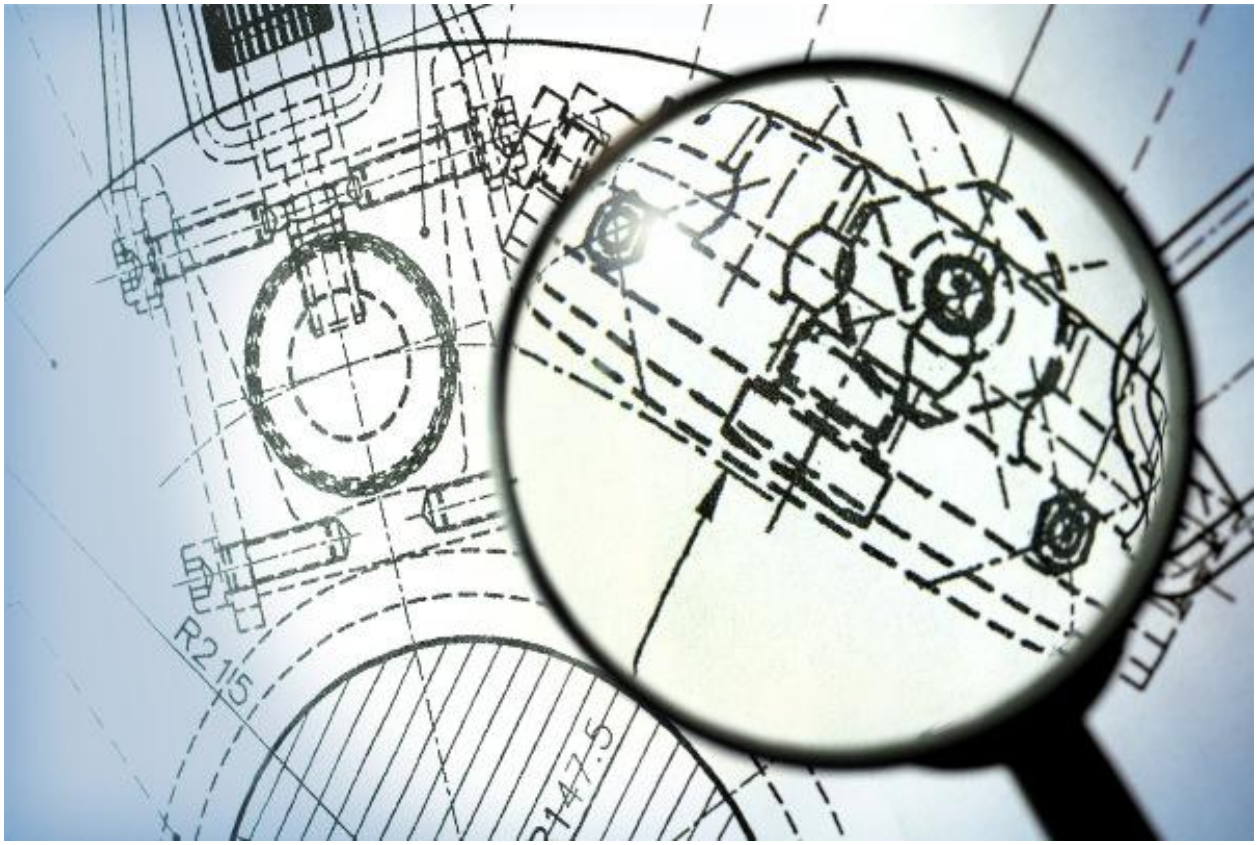
User Manual

Sybit Model Tester



The contents of this document are intellectual property of Sybit GmbH. The document or parts of the document may only be published or passed on to third parties with the permission of Sybit GmbH.

Sybit GmbH
Sankt-Johannis-Str. 1-5
78315 Radolfzell, Germany

Phone: + 49 (77 32) 95 08-0
Fax: + 49 (77 32) 95 08-111

info@sybit.de
www.sybit.com

Dokumentenart	Benutzerhandbuch EN
Dokumenttitel	Sybit Model Tester
Dokumentuntertitel	Benutzerhandbuch
Verfasser	Sybit GmbH Sankt-Johannis-Str. 1-5 78315 Radolfzell
Dateiname	SMT_Benutzerhandbuch_EN_02.02.doc
Model-Tester-Version	2.2
Dokumentenversion	EN_02.02
Status	Freigegeben
Freigabedatum	20.04.2011

Contents

1 Introduction 5

2 Basic Concepts 5

2.1 Example 6
2.2 Test Case and Test Run 6
2.3 Testing Prices 8
2.4 Creating and Maintaining Test Cases in the Sybit Model Tester 9
2.5 Test Cases for Multi Level Configuration Models 10
2.6 Executing Test Cases in the Sybit Model Tester 10
2.7 Profile 10
2.8 Partial Tests vs. Complete Tests 10
2.9 Test Run in VC and IPC 12
2.10 Test Suite 12
2.11 SAP Engineering Change Management (ECM) 13

3 First Steps 14

3.1 Create a Test Case 14
3.2 Run the Test Case 14
3.3 Evaluate 15
3.4 Further Steps 15

4 Reference 16

4.1 Usage of Tables and Other User Interface Elements 16
4.2 Common Header Area 18
4.3 Common Footer Area 23
4.4 Navigation, Loading, and Personal History 23
4.5 Common Selection Component for Test Cases and Test Suites 25
4.6 Welcome 32
4.7 Import 33
4.8 Test Case 37
4.9 Test Suite 48

4.10 Test Run 50
4.11 Evaluation 51
4.12 Mass Maintenance 61
4.13 Profile 70
4.14 Job 77
4.15 Saving Test Cases from CU50 82

5 Error Handling 84

5.1 Termination (Dump) of the Application 84
5.2 Wrong Functionality 84

Appendix 86

A. Messages 86

A.1. VC-Specific Messages 86
A.2. Errors in the Test Case Structure 87
A.3. Error Messages Caused by Testing Prices 87

B. Further Information 88

1 Introduction

This document serves as a user manual and technical reference for users of the Sybit Model Tester (SMT). It is organized as follows.

Chapter 2 covers the basic concepts of the tool, such as test case and test run. *Chapter 3* contains a short tutorial that takes only a few minutes.. It shows – on a beginner level – how to create, execute, and evaluate a test. *Chapter 4* contains a detailed reference of all available functions. Finally, *Chapter 5* tells you what to do, if you encounter malfunctions in the application.

SMT 2.2

In this manual, features that have been introduced or fundamentally changed in this version are marked with a label on the side margin. An example is displayed here.

2 Basic Concepts

The Sybit Model Tester is a workbench to create, save, organize, and execute test cases in the SAP configuration area. It replaces manual testing – e.g. via transaction CU50 – to a large extent. The basic idea of the Sybit Model Tester is to save test cases and to make them automatically reproducible at any time.

Hence, the central element of Sybit Model Tester is the **test case**.

A test case consists of two parts. The first part is the **input data.** Here, you specify the user input for the test case. The second part is the **expected result**. The expected result contains the user's expectation on what the variant configurator (in the following "VC" or "LO-VC") should do based on the input data.

If you run a test case – which is then called a **test run** – the Model Tester hands over the input configuration to the LO-VC. Subsequently, the LO-VC computes a result which we call the **factual result**. The Model Tester automatically compares the expected with the factual result. If the two results match then the test run is successful (denoted by a green light). If the expected result differs from the factual result then the test run fails (denoted by a red light). The behavior of test runs (i.e., its evaluation) can be influenced by means of so-called **profiles**.

Fig. 2-1 (a) shows the traditional way of manual testing via CU50. The tester manually compares the factual result that is computed by LO-VC with the testers expected result. Fig. 2-1 (b) shows the scheme in a Sybit Model Tester scenario. Here, the expected result is stored inside the test case and – hence – can be compared automatically with the factual result.

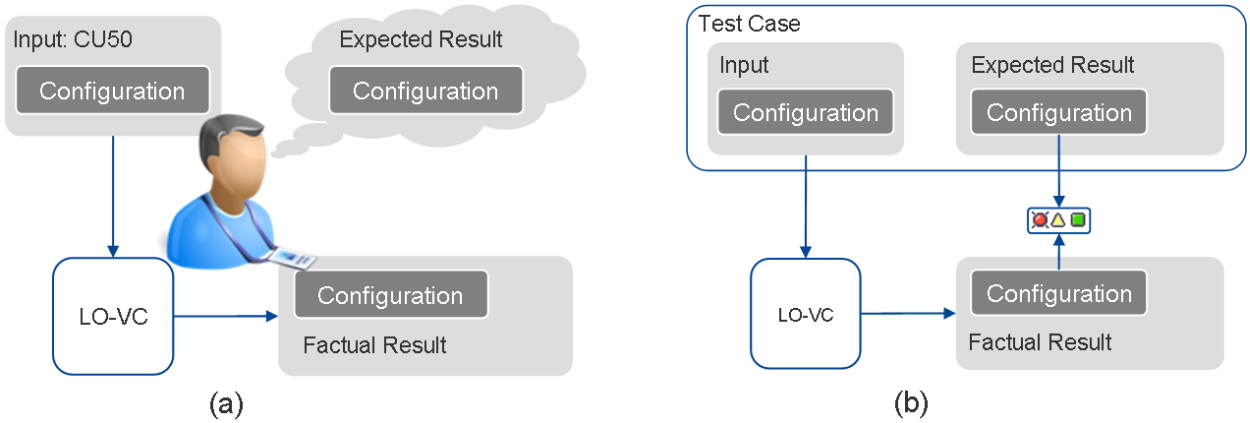


Fig. 2-1: Testing of configuration models. (a) Manual testing. (b) Scenario with the Sybit Model Tester.

2.1 Example

A product has characteristics for size and color. The configuration model specifies that very large instances of the product (`Size = XXL`) should only be available in red (`Color = Red`). Hence, if you input `Size = XXL` the model should automatically infer `Color = Red`. A test case for this behavior would look like this:

Input configuration: `Size = XXL`

Expected result:
```
Size  = XXL
Color = Red
```

The expected result also contains the value `XXL` for characteristic `Size`, since the LO-VC does not delete the value.

When the test case is executed, the LO-VC generates a factual result that is recorded by the Model Tester. The following table shows several cases of potential factual results and how the Model Tester deals with them.

Factual Result	**Comment**
`Size = XXL` `Color = Red`	Factual result matches the expected result: Test run successful, green light
`Size = XXL` `Color =`	Color has not been set: Test run not successful, red light
`Size = XXL` `Color = blue`	Color is wrong: Test run not successful, red light
`Size =` `Color =`	XXL could not be set: Test run not successful, red light

Table 2-1: Alternatives of different factual results for the example.

2.2 Test Case and Test Run

A test case covers checks different aspects of a configuration model.

- **Error and warning messages**: An important use case is to find out if certain configuration instances are prohibited by the configuration model. In the manual testing scenario, these

conflicts are displayed inside a popup dialogue. In the Sybit Model Tester, the expected result contains an area where you can specify error and warning messages. If an expected message is not generated by LO-VC, i.e. if the factual result does not contain the message, the test run fails.

- **Configuration**: The most frequent test type checks if the configuration model sets valid characteristic values. On input of a configuration (i.e. a set of characteristics and values) the LO-VC computes an inferred configuration that is deduced from the input configuration by means of the configuration model. To this end, the Model Tester's expected result contains an area to specify an expected configuration.
- **Bill of materials (BOM)**: The expected result contains an area to specify the expected bill of materials, to test the BOM explosion.
  The bill of material may contain the following types of positions:
  - Stock item (L)
  - Non-stock item (N)
  - Document item (D)
- **Variant conditions / prices**: If there are errors in pricing, financial damage is often unavoidable. You can test condition records that are controlled by variant conditions in the prices area.

Fig. 2-2 gives an overview including the test case (with input configuration and expected result), the LO-VC, and the factual result.

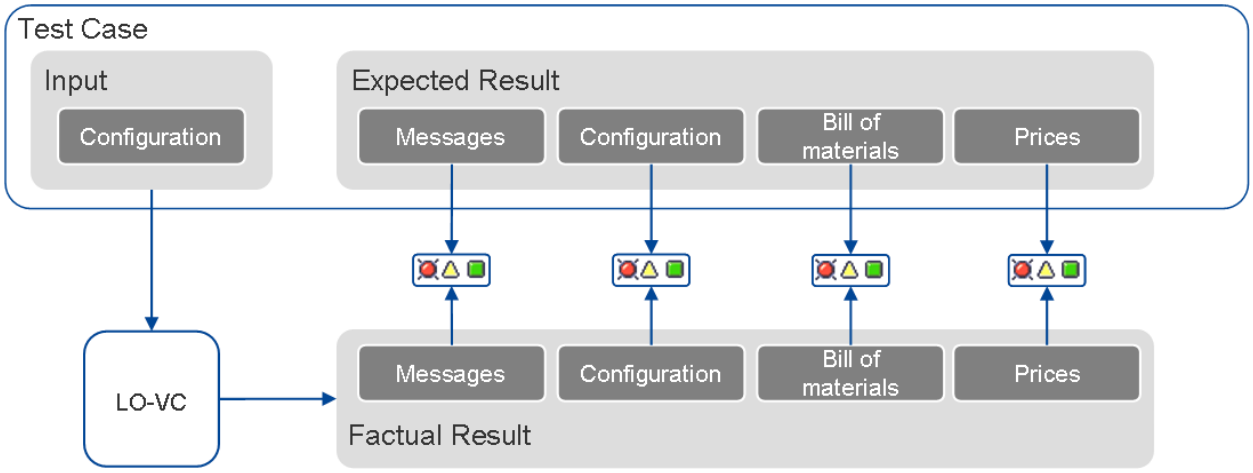


Fig. 2-2: A test case consists of an input configuration and an expected result. When a test is run, the input configuration is passed to the LO-VC that generates a factual result. Finally, the Model Tester compares the factual result with the expected result.

The comparison between expected and factual results is executed for each area separately, i.e. messages, configuration, bill of materials and prices. To give a quick overview of these results, the Sybit Model Tester uses icons that have the colors of traffic lights. The meaning of the colors is as follows:

- **Red light:** The test run produced an error. The factual result does not match the expected result.
- **Green light:** The test run had no error. The factual result matches the expected result.

- **Grey light**: Nothing has been tested. This case occurs in the following cases:
  - The area has been marked as irrelevant by means of a profile.
  - If a test case header contains no calculation scheme, the prices area is marked grey.

Positive vs. Negative Tests

In principle we distinguish between positive and negative tests. A positive test expects a valid configuration. In the Model Tester a positive test checks system set characteristic values, the explosion of the bill of materials (BOM), and pricing information for relevant condition records.
In contrast, a negative test checks if a certain configuration is prohibited by the configuration model. In the Model Tester you can realize this kind of test by expecting error and / or warning messages. Usually it does not make sense to test system set values, BOM explosion, or prices in a negative test.

2.3 Testing Prices

The prices area in the test case is different from the other areas since the information stored in this area is not available at any other place in the standard SAP. Where system set values, BOM explosion, and error or warning messages can be assessed directly via CU50 or inside a sales document, the prices area contains all condition records that are relevant for any variant condition for a given configuration.

To understand the testing of pricing better, we first have a quick look at the Model Tester for relevant aspects of pricing in SAP (see Fig. 2-3). There, a calculation scheme contains a number of condition types. Each condition type specifies a sequence (10, 20,) of accesses, the so-called access sequence.

An access is basically a table that has condition records as rows. This table has at least the following columns:

- **A**: Amount
- **C**: Rate unit (Currency or percentage)
- **M**: Condition unit
- **p**: Condition pricing unit

Additionally a sequence may define a number of keys (i.e. columns of the table) that are used to determine a specific condition record and its price. A typical key column is e.g. the variant condition which is used to bind prices to specific configurations. In an access sequence, accesses are always ordered from specific to general, i.e. with increasing access numbers, accesses get less specific. Hence, the last access contains the smallest number of key columns.
In the example displayed in Fig. 2-3 access 10 of condition type 1 contains four columns for sales organization (SalesOrg), distribution channel (DistChan), material (Material), and variant condition (Variant). In contrast, access 20 only contains columns for material and variant condition.

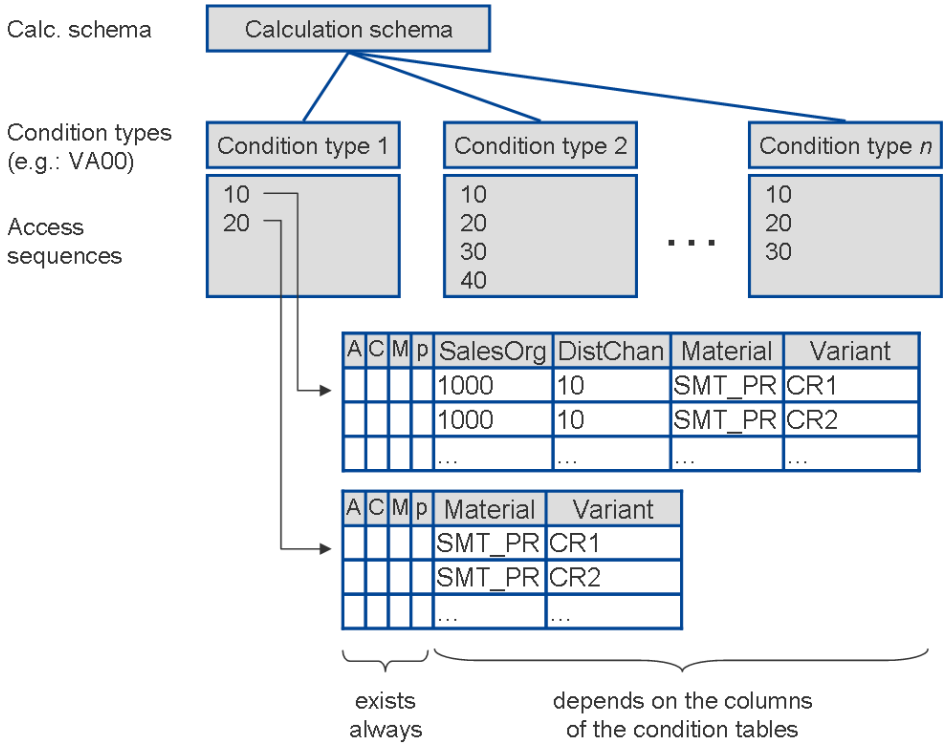


Fig. 2-3: Excerpt from structures relevant to pricing.

The Sybit Model Tester checks all condition records that are relevant to the calculation scheme specified in the test case. Here, relevance is always determined by variant conditions that result from characteristic values in the input configuration of the test case. For all condition types of the calculation scheme, all access sequences and, hence, all accesses are evaluated. This results in a list of all relevant condition records.

2.4 Creating and Maintaining Test Cases in the Sybit Model Tester

To minimize the effort for creating test cases the Model Tester offers multiple ways to create test cases.

- CU50-Save: An extension of transaction CU50 enables you to save configurations directly from the transaction into the Model Tester. Please check with your administration if and on which systems this functionality is available. We describe the functionality in Detail in Sec. 4.15.
- Import from sales documents: Each inquiry, quotation, and sales order contains valuable information on configuration and sales relevant BOM positions of the respective material. The Sybit Model Tester reads this information and generates test cases fully automatically. We describe this function in detail in Sec. 4.7.1.
- Import from Excel / XML: In companies that already have a formal model verification / test process, test cases are often stored in Microsoft Excel files. The Model Tester can import these test cases into its test cases database via the XML upload interface. We describe this function in detail in Sec. 4.7.2.
- Manual editing: Besides the possibilities to import test cases, the Model Tester contains a tool to create and edit test cases directly via the graphical user interface. We describe this function in detail in Sec. 4.8.

If you change your configuration model, you have to adapt your test cases accordingly. The tool for test case mass maintenance offers you the possibility to adapt a large number of test cases simultaneously and with a few clicks only. E.g. if you inserted a new required characteristic into you configuration model you may open any number of test cases in the mass maintenance tool and add a value for this characteristic to all test cases. This function we describe in detail in Sec. 4.12.

2.5 Test Cases for Multi Level Configuration Models

The Sybit Model Tester can test multi level configuration models.

In a multi level configuration model people usually distinguish the material which is on top of the material hierarchy. This material is called **top material**, **header material**, or **root instance**. Materials further down in the hierarchy are called **subordinate material**, **component,** or **child instance**.

In the Sybit Model Tester, a test case for a multi level configuration model contains the information for materials on all hierarchy levels. By means of a navigation tree you may switch between hierarchy levels. Each test case in the Sybit Model Tester relates to its respective material. Multi level test cases always relate to their root instance.

To distinguish between the different hierarchy levels inside a test case, we use the following terms:

- Root-Item: The root-item of a test case contains all information that is necessary to test the root instance material.
- Sub-Item: Sub-items of a test case contain the information that is necessary to test one specific subordinate material.

2.6 Executing Test Cases in the Sybit Model Tester

In the Sybit Model Tester you may trigger test runs directly by pressing a button. This function is described in Sec. 4.10. As an alternative, you may also schedule test runs via the standard SAP job scheduling mechanism. This enables you to execute test runs in a background job in a periodical schedule. E.g. you may execute regression tests every night fully automatically. We describe this function in detail in Sec. 4.14.

2.7 Profile

In the Sybit Model Tester, you may use **profiles** to fine tune the behavior of test runs. Depending on the usage scenario it might be useful to disregard certain information from the configuration area or the BOM area, i.e. if an error occurs in an irrelevant part, the user should not be informed. In the profile you may specify which areas should be relevant for the evaluation. In particular you may even define that you are not interested in the values of certain characteristics. This is helpful, for example, for invisible characteristics that are only used for internal purposes. In the BOM area you may define whether the evaluation should use all available information or if, i.e., inconsistencies in position numbers or units should not be considered relevant to the user. If you don't use a profile for a test run, all available information is considered relevant. Profiles are discussed in detail in Section 4.13.

2.8 Partial Tests vs. Complete Tests

The Sybit Model Tester focuses on the execution of regression tests of configuration models – i.e. periodically recurring test runs. Hence, it supports the creation and maintenance of configuration models over their entire life cycle. To realize effective quality control, you have to make sure that the following two aspects are met:

1. All values from the expected result are contained in the factual result.
2. All values from the factual result are contained in the expected result.

In the Sybit Model Tester, the standard test case tests both aspects. The run of such test case is successful if and only if the expected result exactly matches the factual result. (Here, we neglect effects that you may cause using profiles – for this, please refer to Section 2.3.) In the Sybit Model Tester, this standard test type is called **complete**. It can be identified by the following icon:

However, during the implementation phase of a configuration model we often have the situation where we want to test *specific aspects* of a configuration model, although not all aspects work correctly. To support this kind of testing, you can define **partial** tests in the Sybit Model Tester. You can identify partial tests by means of the icon. Partial tests only cover the first of the two above aspects, i.e. if all values from the expected result are contained in the factual result. If there are values in the factual result that you did not expect, the test run is marked successful (i.e. green icon) anyway.

Table 2-2 summarizes the differences between a test run of a partial test and the test run of a complete test.

Area	**Partial Test Case**	**Complete Test Case**
Messages	- Only those messages are shown that affect characteristic values contained in the input configuration. - No messages from structure conflicts in multi-level configuration - No messages concerning variant conditions that are not contained in the expected result.	All messages are shown.
Configuration	- Characteristic values in the factual result that are missing in the expected result **are not** considered as a failure. - The factual result only contains characteristic values that are contained in the expected result.	- Characteristic values in the factual result that are missing in the expected result **are** considered as a failure. - The factual result contains **all** characteristic values.
Bill of Materials (BOM)	- Missing entries in the expected result **are not** considered as a failure. - The factual result only contains entries that are contained in the expected result. For the comparison between expected and factual result, all properties (that are not marked as irrelevant by the profile) of the BOM line are used.	- Missing entries in the expected result **are** considered as a failure. - The factual result contains **all** entries.
Pricing	- Missing condition records in the expected result **are not** considered as a failure. - The factual result only contains condition records that are contained in the expected result.	- Missing condition records in the expected result **are** considered as a failure. - The factual result contains all condition records.

Table 2-2: Differences between partial and complete test cases

2.9 Test Run in VC and IPC

The Sybit Model Tester may execute test runs in both, the (LO-) VC and the IPC. Please notice the following differences:

- **Pricing system**: In the current software version, there is a difference in testing prices / condition records in VC and IPC. In the current software version the Model Tester can not connect to CRM systems. Hence, to check the prices area of a test case, it connects to an according ERP system. To find out which ERP system is assigned to the IPC system defined in the header of your application, please contact your administrator.
- **Prices area – factor irrelevant**: IPC and VC behave differently when calculating factors for condition records. Hence the factor column is always marked irrelevant in test runs on the IPC. I.e., if expected and factual factor do not match, this will not cause the test run to fail.

Specifically when processing invalid configurations, the IPC behaves differently from the VC. Where the VC stops processing the configuration model after certain conflicts have occurred, the IPC continues its model evaluation. As a consequence, the Model Tester distinguishes a certain kind of negative tests (for an explanation of positive and negative tests, see the info box in Sec. 2.2). The following kinds of negative tests are affected:

- **Characteristic inconsistent**: The messages area contains a message of type “Characteristic inconsistent”.
- **Value inconsistent**: The messages area contains a message of type “Value inconsistent”.

For those test cases there are the following differences:

- **Test case partial**: When the VC hits an error / inconsistency in the configuration, it stops evaluating the configuration model. In contrast, the IPC continues the evaluation. Hence, the IPC returns a more complete, but at the same time different result from the VC. The result of the IPC contains the result of the VC. The Model Tester ensures that a negative test case that is valid for the VC is also valid for the IPC. To this end, it automatically switches to the partial evaluation logic in these cases.
- **Messages area – constraint irrelevant**: In contrast to the VC, the IPC does not return the name of constraints in case of occurring messages. Hence, the constraint name of messages is always regarded as irrelevant. Hence, if the constraint name in the expected result does not match the constraint name in the factual result, the test run gets a green icon anyway. In the analysis view the constraint name is can be displayed, however it is marked as irrelevant (i.e.: dark blue background).
- **Prices area** – factor irrelevant: IPC and VC behave differently concerning the calculation of the factor in condition records. Hence, for the above specified types of negative tests the factor is marked as irrelevant in IPC test runs. If a factor in the expected result differs from the according factor in the factual result this will not lead to a conflict (red icon).

2.10 Test Suite

To simplify the management of test cases, they can be organized in test suites. A **test suite** is a container that may contain any number of test cases. Those test cases can be of any type, see Section 2.8. One single test case can be contained by multiple test suites. Test suites are described in detail in Sec. 4.9.

2.11 SAP Engineering Change Management (ECM)

Until version 1.5, the Sybit Model tester could be used in two different variants which were targeted towards enterprises with and without the SAP Engineering Change Management (ECM). The variant without ECM was dispensed with in version 2.0. If you want to get rid of the according elements in the user interface, you can do this by means of the Web Dynpro standard personalization functionality. We are happy to support you in this respect, see Appendix B.

3 First Steps

This chapter contains a first tutorial. Here you can create, execute and evaluate your first test. We expect the tutorial to take you about 10 minutes.

3.1 Create a Test Case

- **Open**: Open the Sybit Model Tester
- **Insert KMat**: In the header area, there is a field labeled **KMat** that contains the ID of the configurable material under test. You may enter a material directly or use the value help. To begin with, you may want to choose a simple material with only few required characteristics.
- **Test case**: Navigate to the tab **Test Case**.
- **Create**: Click on **New**.
- **Name / Description**: Insert a name and a description for the respective input fields, i.e. “first test” and “Tutorial”.
- **Input configuration**: The main work area consists of a navigation table and two areas for input configuration and expected result. The table for input configuration should contain all characteristics of the chosen KMat. Please enter a valid configuration into the table (similar to CU50).
- **Expected Configuration**: The area on the right contains the expected result. Switch to the tab Configuration.
- **Generate Proposal**: In the configuration table of the expected result you could enter the expected configuration manually. A more convenient way to create the expected result is to let the LO-VC generate the result and only interfere if this is not what you actually expect. To achieve this, take a look at the toolbar above. Besides the label **Expected Result** there is a checkbox for each area of the expected result. Check the check boxes for **M** and **C** and click on **Generate Proposal**. Now the LO-VC fills the message and the configuration area of the expected result with the output of the current configuration model. If your input configuration is invalid you will see a warning inside the message area at the very bottom of the screen. If you need details on the errors just switch to the messages tab of the expected result. Then, correct your input configuration and press generate proposal again. Repeat these steps until you have no messages left in the message tab.
- **Save**: Click on the save button.

3.2 Run the Test Case

- **Test Run Tab**: To switch to the Test Run tab you may use the **Quick Link** at the lower right of your screen. You are taken directly to the test run tab where your test case is already loaded and selected inside the selection table. As an alternative you can also click on the test run tab and search for your test case in the data base.
- **Execute**: Click run. On successful execution you will see a message inside the message area.

3.3 Evaluate

- **Evaluation Tab**: Switch to the evaluation tab once again by using the **Quick Link** at the lower right of your screen. The executed test run is already loaded and selected in the evaluation table. As an alternative, you may also click on the evaluation tab directly and load your test run.
- **Analysis**: A click on the **Analysis** button opens the analysis popup. You can see the expected result on the left and the factual result on the right hand side for all areas. If everything went o.k. you should only see green or grey lights only.

3.4 Further Steps

- **Test case**: A click on the name of the test case brings you back to the test case tab. There you may change the input configuration or the expected result.
- **Test run**: Switch to the test run tab and execute your test case.
- **Evaluation**: Check out how your changes affected the result of the test run.

4 Reference

The user interface of the Sybit Model Tester consists of a number of tabs that share a common header and footer area. In this section we first present the header and footer area and a common user interface component to select test cases and test suites. Subsequently, we go into detail for each single tab.

4.1 Usage of Tables and Other User Interface Elements

The Sybit Model Tester uses the Web Dynpro Abap technology to realize its user interface. Hence, the entire Web Dynpro Abap functionality is available for Model Tester users. Especially the following elements are worth noticing:

4.1.1 Selection of Table Lines

As everywhere in the SAP system you have the following possibilities to select table lines. Please note that the selection of multiple lines – as described in point 2 and 3 – is not enabled everywhere in the application.

- Single line: With a mouse click (left click) on the button in the selection column[1] of the table you select exactly one line. The selection column is the first column of each table.
- Multiple, consecutive lines: You can select multiple, consecutive lines as follows: First click on the selector of the first line. Then hold down the Shift-Key (Arrow up) and click on the selector of the last line. This also selects all lines in between. You may also scroll the table between selecting first and last line.
  You may also select the lines vice versa, i.e., you select the last line first and then the first line.
- Multiple lines: You can select multiple lines that need not be adjacent by holding down the Ctrl key while clicking on selectors. You may scroll the table between clicks.
- All lines: Via a click on the upper left corner of a table you may chose to select all lines of a table.
- Delete selection: Via a click on the upper left corner of a table you may chose to de-select all lines of a table.

4.1.2 Dealing with Table Columns

To ease the handling of tables with many columns the following two mechanisms are available:

- **Horizontal scrolling**: If the application has not enough space to display all table columns the technology automatically displays pushbuttons for scrolling or a scrollbar at the footer of the table. With these elements you may display table columns that are “to the right” of the displayed table columns.
- **Hiding columns**: On a personalization basis you may hide specific columns. To achieve this, right-click on the table column in the header. This opens a context menu, see Fig. 4-1. Choose **User Settings → Hide Table column** to hide the respective column. To display a column that has been hidden previously, choose **User Settings → Invisible Elements →**

[1] http://help.sap.com/saphelp_nw70/helpdata/en/b5/ac884118aa1709e10000000a155106/content.htm

Restore Table Column.
The application recognizes hidden table columns as part of the personalization. At the next start of the application, the system will still hide this column. The interface element can be configured individually at each location within the application. E.g. in the evaluation you may want to see the result of the last run of a test cases, whereas in the test case view you may want to see which user did the most recent changes on the test case.

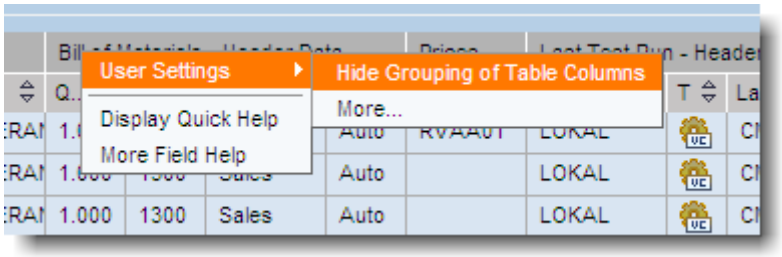


Fig. 4-1: Hiding of table columns via the context menu.

- **Sort columns**: By means of the same mechanism that helps you to hide table columns you can also reorder them. You just have to click on More… in the context menu of the element that comprises the column, see Fig. 4-1. For a column this may either be a column group that contains the column or the table. In the latter case you have to right-click on upper left corner of the table. This triggers a popup where you can sort the columns of table on the right by moving them up or down, see Fig. .

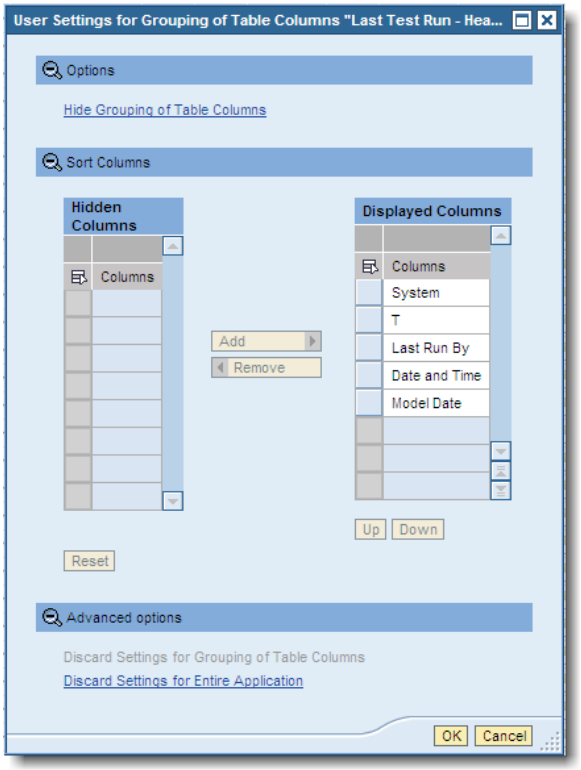


Fig. 4-2: Sorting table columns.

4.2 Common Header Area

The header contains a number of fields that are relevant for all tabs, see Fig. 4-3.

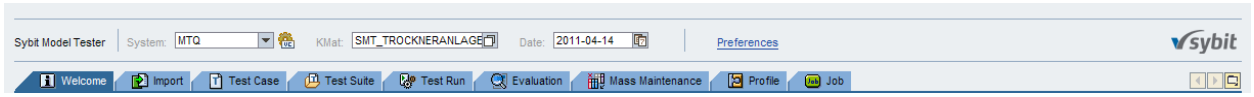


Fig. 4-3: Common header area.

4.2.1 System, KMat, and Date

If your enterprise specific application configuration contains more than one system to test, you will see a drop down box with the available scenarios.

Via the system drop down box you do not only select the physical system, but you also set the configurator type, i.e. VC or IPC where applicable. For the difference between VC and IPC please also refer to Sec. 2.9. An according VC icon ( ) or IPC icon ( ) is displayed right beside the drop down menu. If you chose a VC system, the following actions are executed on the system:

1. Reading the master data
2. Test runs (messages, configuration, and bill of materials)
3. Test runs (determining condition records)
4. Import of sales documents

In contrast, if you select an IPC system, only point 2 is executed in the IPC (in the current software version). This has the following consequences:

- Reading master data is not possible if you select an IPC system. This implies that you can not create or change test cases if your system is an IPC system, see Sec. 4.8.
- Testing of condition records is executed in the ERP system that is assigned to the IPC system in the header. Testing condition records in the CRM system is not supported by the current software version.
- Import of sales documents (see Sec. 4.7.1) is executed in the ERP that is assigned to the IPC system. Import from CRM systems is not supported by the current software version.

To find out which ERP system is assigned to your IPC system, please ask your system administration.

The **KMat** field contains the ID of the configurable material (KMat) under test. The **Date** field contains the logical model date. Example: if you enter 08.11.2010 in the date field then your test runs will be executed with the configuration model that is active at that date.

4.2.2 Preferences

If you click on **Preferences** you will get a popup that contains system settings and personal preferences (Fig.). The navigation tree on the left contains the following chapters: **System Information**, **Presentation**, **Application Parameters**, **Testcase Parameters,** and **Statistics and License**.

When you open the popup, the **System Information** chapter is opened (Fig.). It contains four sections. The section **General** gives you information about your user and the system you are working on. In the section **Application Log** you can display the application log and switch it on and off. You should be aware that switching on the application log can have negative impacts on your application performance.

Caution: Switching on the application log may lead to loss of performance.

The section on **System Configuration** contains information on your enterprise specific configuration of the Sybit Model Tester application. Here you can view the RFC connections that are used to connect to the VC and IPC systems available in your scenario as well as a list of users who have administration privileges.

SMT 2.2

Your personal settings which the system will remember until your next session are displayed in the **Personalization** section. Here you also find a push button to delete your personal history. If you click on it, the table in the Welcome screen and the personal history lists besides the Load buttons in all tabs will be deleted.

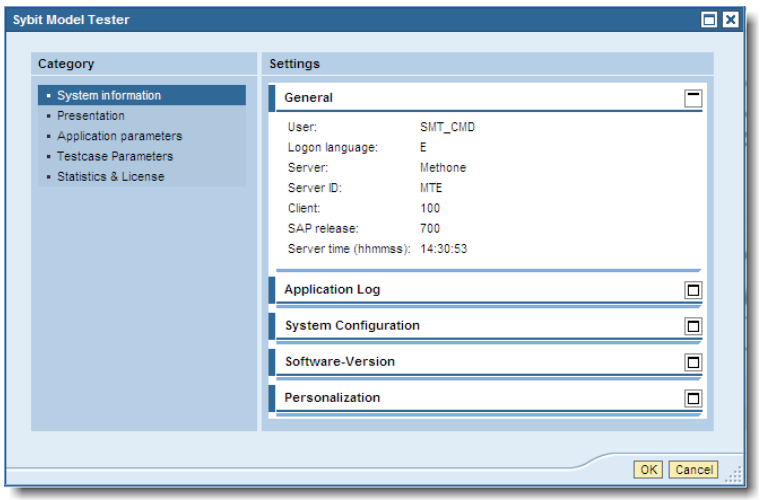


Fig. 4-4: Preferences popup: Chapter System information.

In the chapter **Presentation**, you can switch between language dependent and language independent display of master data (Fig. 4-5).

- Characteristics and Values: This setting affects the presentation of characteristics and their values. If you chose both options, both language dependent and language independent texts are displayed side by side.
- Configuration structure: This setting affects the presentation of KMats in the navigation tables in the test case or the profile tab

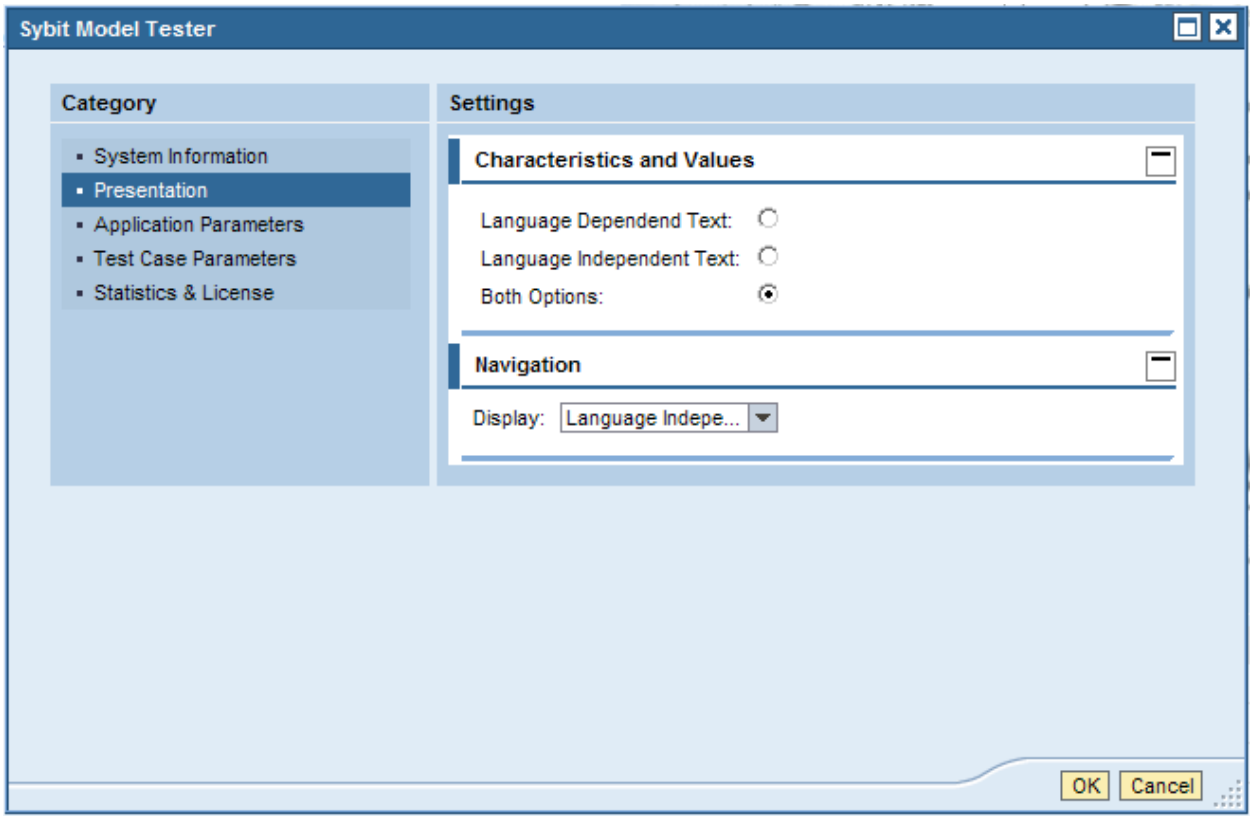


Fig. 4-5: Preferences popup: Chapter Presentation.

You can set the application size in the **Application Parameters** chapter (Fig.). Here you can optimize the layout of the application to your screen resolution. If you click on the Reset Default Values button the application size is reset to the default values of 1200x850 pixels. In the lower section you can customize the application to always set the current date into the Date field in the header, or to remember the date and set it in your next session.

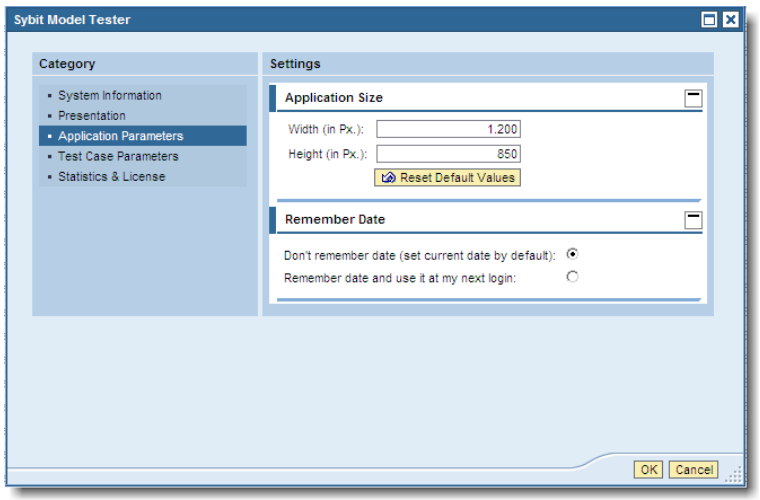


Fig. 4-6: Preferences popup: Chapter application parameters.

You can set the following preferences in the chapter on **Testcase Parameters** (Fig. 4-7):

General Settings:

- If you remove the check from “refresh automatically while ‘generate proposal’”, then the generate proposal button in the test case tab changes its behavior. If the check is set, then after each click on Generate Proposal the test case hierarchy is refreshed automatically. If the check is not set, you have to do this manually. As an illustration, consider the following two workflows that have the same result:

Workflow 1	Workflow 2
• Preferences: Check is set • Test case tab: Click on generate proposal	• Preferences: Check is **not** set • Test case tab: Click on generate proposal • Test case tab: Click on refresh

Default Settings for the Bill of Materials:

- **Plant:** If you create a new test case in the test case tab, then this plant is inserted into the new test case.
- **Usage:** If you create a new test case in the test case tab, then this BOM configuration is inserted into the new test case.
- **Application**: If you create a new test case in the test case tab, then this BOM application is inserted into the new test case. Choose Automatic if you want the BOM application to be determined by the configuration profile. You have to chose Manual and set the according application only if there is no BOM application in the configuration profile.
  This parameter is not only used when creating test cases in the test case tab, but also when you import test cases from sales documents.

Default Settings for Prices

- **Calc. Schema:** If you create a new test case in the test case tab, then this calculation schema is inserted into the new test case. If you don't want to test pricing, just leave the field empty. Then all test cases will be created with empty calculation schema.

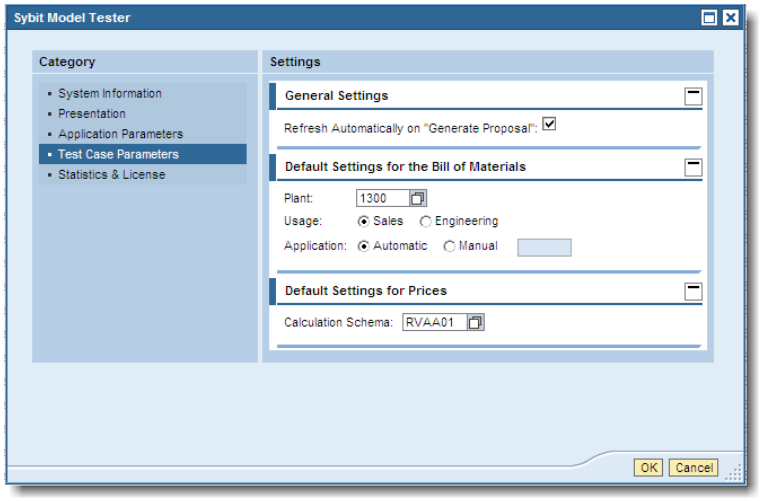


Fig. 4-7: Preferences popup: Chapter Testcase Parameters.

The chapter on **Statistics & License** (Fig. 4-8) provides you with some statistics on the usage of your installation of the Sybit Model Tester. This also includes information about the number of licensed characteristics and how many you already test. Caution! The computation of these statistics causes complex database queries that may take a while.

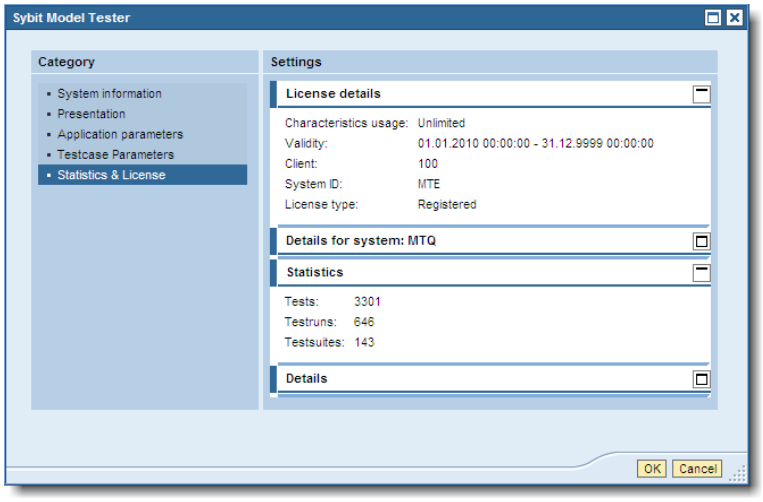


Fig. 4-8: Preferences popup: Chapter Statistics & License.

4.3 Common Footer Area

The footer area comprises two parts (Fig. 4-9).

On the left side you find the message area of the application. Here, different types of messages may occur. Informative messages give a positive feedback on system actions. They come along with a green light. Warning messages are yellow. Red messages occur in cases where the system could not execute out your requested action.

On the right side you find the navigation area containing so-called **Quick Links**. Depending on your actions the system suggests where you could go next. Different from changing the tab, here you can take your context – i.e. the object(s) you are currently working on – with you. Example: if you are viewing a test case inside the test case tab, there is always a quick link to the test run tab. A click on the quick link opens the test run tab. Simultaneously the test case you where just working on is automatically loaded into the selection table such that you can trigger the test run with one single click.

Imported test 'SD_0000000059-000010_007' from sales order '0000000059'.
Imported test 'SD_0000000054-000010_011' from sales order '0000000054'.
Quick Link Load in: Test Run Load in: Test Suite

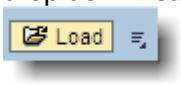
Fig. 4-9: Common footer area.

4.4 Navigation, Loading, and Personal History

In the Sybit Model Tester, navigation and loading of objects can be achieved by different GUI elements.

- **Welcome tab:** In the welcome tab (see Section 4.6) there is a table that displays your recently executed actions that you may wish to resume. These actions depend on the settings in the header area. You will find different actions in this table for different KMats, see Table 4-1. This action list that reflects you personal history can be deleted in the Preferences window in the area System Information → Personlization → Reset Reset personal history, see Section 4.2.2.
- **Click on tabs:** Clicking on tabs is the classical way of navigation. You can do this at any time and in any state of the application. If there are unsaved changes in your current working tab, you will get a popup that asks if you want to save or discard those changes before navigation.

- **Load button:** The following tabs contain buttons to load an object: test case, test suite, evaluation, mass maintenance, and profile. If you click the button, a popup will appear where you can search for and subsequently load found objects. Search results depend on the information on KMat, date, and System that are specified in the header, see Table 4-1.
- **Personal history:** In the following tabs you may load objects from your personal history: test case, test suite, evaluation, profile, and job. To do this, click on the small symbol for drop down lists beside the load button.



Objects shown here depend on the information on KMat, date, and system that are specified in the header, see Table 4-1. The personal history can be deleted in the Preferences window in the area System Information → Personlization → Reset Reset personal history see Section 4.2.2.

- **Navigation area in the footer**: In many tabs, you will get navigation helps by means of quick links in the footer area, see Section 4.3.

Tab	System	KMat	Date	Explanation
Welcome				
Test Case		X	X	Only the following test cases are found: - they test the given header KMat - their validity time span contains the header date.
Test Suite		X		Test suites are specific to KMats
Test Run	(X)	X	X	The test run is executed in the system that is specified in the system. If you search for test cases, KMat and date are respected.
Evaluation	(X)	(X)		Load: As a default, the system and the KMat are respected, but you may change this in the load dialogue. History: System and KMat are not regarded.
Mass Maintenance		X	X	Same as in Test Case
Profile				Loading profiles disregards data from the header.
Job				Loading jobs disregards data from the header.

Table 4-1: Influences of the information in the header on loading functionality in each tab. The system can be changed in the header only if you have configured multiple target systems in your scenario.

4.5 Common Selection Component for Test Cases and Test Suites

There is a common visual component to select test cases and test suites at various places throughout the application, see Fig. 4-10. In the current version 2.1 of the software, this component has been redesigned and enhanced fundamentally.

The component comprises two tabs labeled with **Tests** and **Suites.** You can search and filter test cases directly in the tests tab. You may use the suites tab to take advantage of test suites as a means to organize test cases. Example: you can search for test suites and, subsequently, you may choose single test cases from within the suites. Since the functionality in the suites tab is similar to the tests tab, we only discuss the tests tab.

Both tabs contain a filter area on the left and a table to display all objects that comply with the filter criteria on the right hand side. You may adjust the table on the right according to your needs, see Sec. 4.1.2.

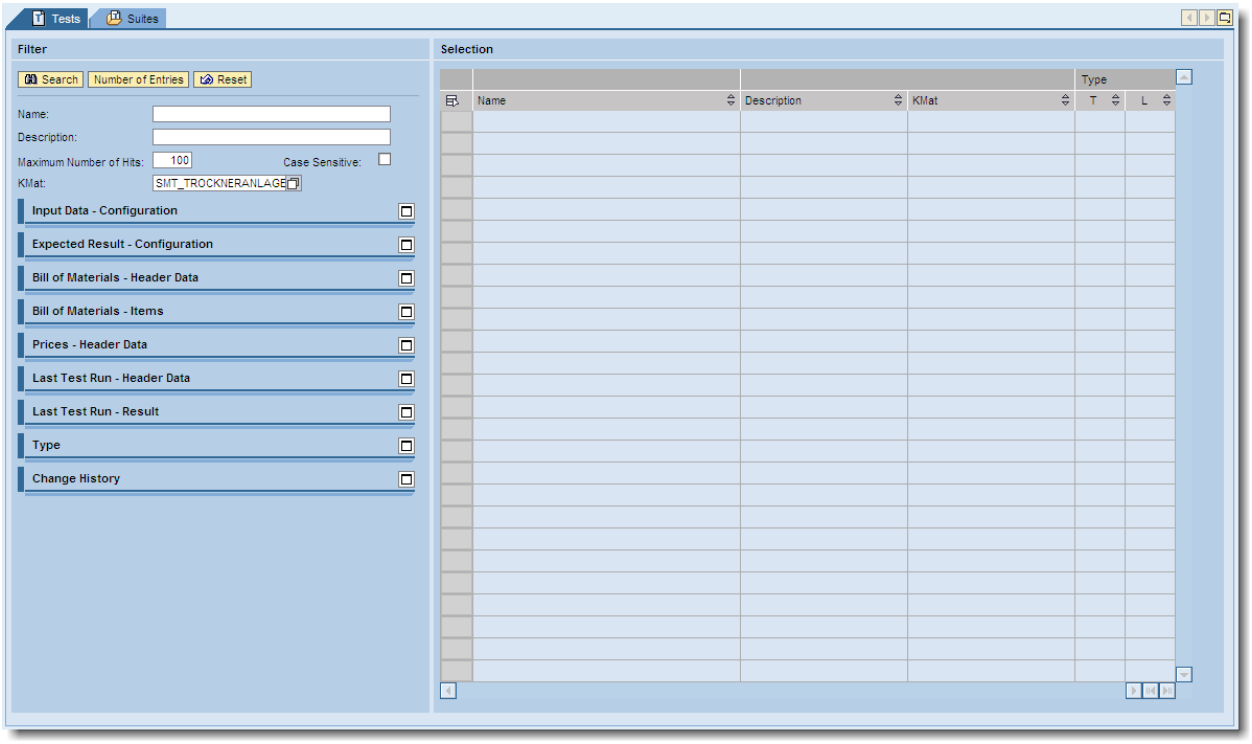


Fig. 4-10: Selection of test cases and test suites.

4.5.1 Filter Area

The filter area consists of a toolbar with buttons, a standard filter area, and a number of expandable filters for extended search possibilities. To enable you to find non-empty filter entries even if their containers are not expanded, the Model Tester marks those containers by means of a star icon (*). All filter entries are implicitly connected by a logical AND operator. I.e. you will get exactly those test cases that match all entered filter criteria.

The toolbar contains the following pushbuttons:

- **Search**: Applies all the below entered filter criteria to all test cases that are currently stored in the test case database. Subsequently it loads all matching test cases into the table on the right.

- **Number of Entries**: Applies all the below entered filter criteria to all test cases that are currently stored in the test case database and counts the number of matches. It does not load anything into the table on the right.
- **Reset**: Deletes all entered filter criteria.

In the filter area you may filter for the following aspects / criteria:

- **Name**: Name of the test case.
- **Description**: Description of the test case.
- **Maximum Number of Hits**: If you click the search button, not more test cases are displayed than specified here. This reduces the time necessary for loading a large number of test cases. If you enter zero ('0') here, all test cases are displayed.
- **Case Sensitive**: If you want to distinguish small and capital letters when searching for name and / or description you have to set this check.

If you open the area **Input Data – Configuration**, you may filter for characteristic values in the input configuration of test cases, see Fig. 4-11. That way you may search test cases with any specific combination of characteristic values. The following controls are available:

- New : Deletes all lines and creates one new, empty line.
- AND : Adds a new line connected with AND at the end of the table.
- OR : Adds a new line connected with OR below the selected line.
- Delete : Deletes the selected line.

The functionality of the buttons ensures that the brackets in the logical expression are always set correctly in terms of the conjunctive normal form[2]. This implies that the expression always consist of AND connected terms where each term is a connection of OR sub-terms.

Example: You want to search for test cases with characteristic values (COLOR = WHITE and SIZE = XXL) or (COLOR = GREEN and SIZE = XXL). Then the equivalent logical expression in conjunctive normal form is (COLOR = WHITE OR COLOR = GREEN) AND SIZE = XXL.

If you set a check in the NOT column then you will get all test cases that do not contain the respective characteristic value. Caution: This kind of search potentially finds a very large number of test cases and may take a while.

Entries in the table cells are always the language independent IDs of characteristics and values, see Table 4-2. However, via the value help you may also search for language dependent names.

If you leave empty the table cell for the characteristic name this is equivalent to a wildcard, i.e. it matches all characteristics in a test case. If you leave empty the cell for the value, this is also treated as a wildcard, i.e. you will get all test cases where the given characteristic has any non-empty value. Lines without characteristic name and without characteristic value are not permitted.

[2] For a detailed description of the conjunctive normal form, please refer e.g. to:
http://en.wikipedia.org/wiki/Conjunctive_normal_form

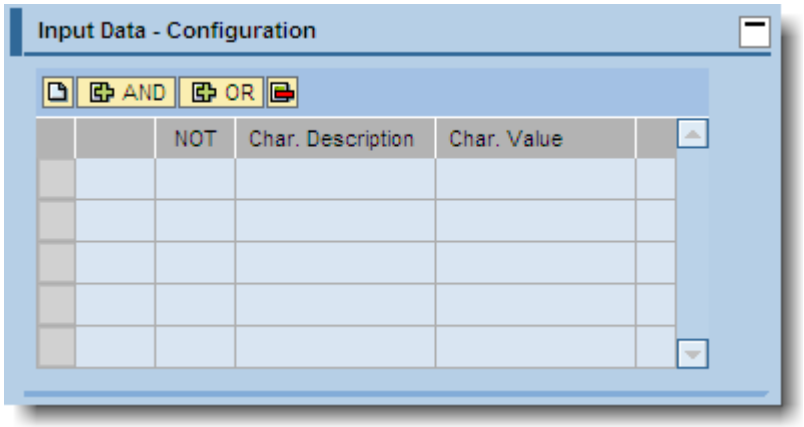


Fig. 4-11: Filtering for characteristic values in the input configuration.

If you open the area **Expected Result – Configuration**, you may filter for characteristic values in the expected result of test cases. This function is similar to the filter in the Input Data – Configuration area but affects the expected result instead.

If you open the area **Bill of Materials – Header Data**, see Fig. , you may filter for the following aspects:

- **Quantity**: The quantity that is stored in the test case and used for BOM explosion.
- **Plant**: The plant used for BOM explosion.
- **Usage**: The usage of the BOM (sales or engineering).
- **Application**: The BOM application stored in the test case.

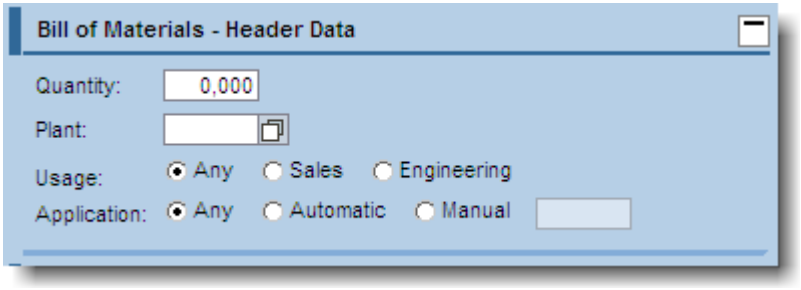


Fig. 4-12: Filtering for header data of the bill of materials.

If you open the area **Bill of Materials – Items**, see Fig. 4-13, you may filter for BOM items that are part of the expected result of test cases. This functionality is similar to the filter for Input Data - Configuration. The difference here is that in each line of the table you may specify a position (**P**), a quantity (**Q**), a unit (**U**), and a component name. Empty entries are always treated as wildcards, i.e. they match all values.

Example: Let's have a look the following filter for BOM items:

	NOT	P	Q	U	Component	
(		0320			SCREW_123K45	)

This filter finds all test cases that have the component SCREW_123K45 at position 0320 in their BOM area in their expected result. The quantity and the unit are treated irrelevant, i.e. the filter finds both, test cases with e.g. quantity =1 and test cases with e.g. quantity =2.

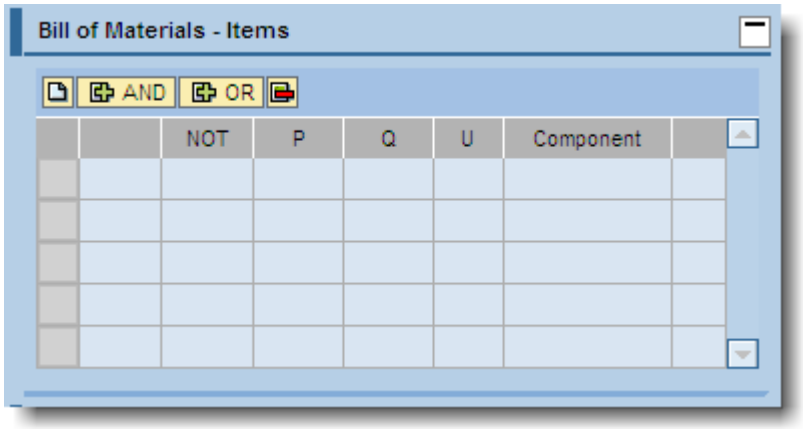


Fig. 4-13: Filtering for items in the bill of materials.

If you open the area **Prices – Header Data**, see Fig. 4-14, you may filter for header information for pricing:

- **Calculation Schema**: The calculation schema of the test case

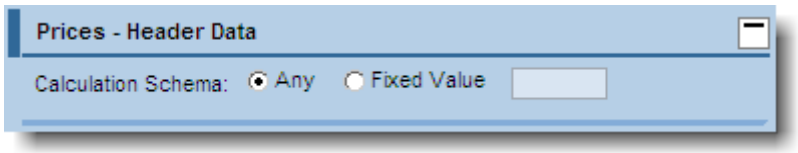


Fig. 4-14: Filtering for header information for prices.

If you open the area **Last Test Run – Header Data**, see Fig. 4-15, you may filter for header data of the last run of a test case. The following criteria are available:

- **System**: The system that has been used to execute the test run.
- **Last Run By**: The user name of the user who triggered the run.
- **Last Test Run At: From / To**: The time span in which the run has been triggered.
- **Model Date: From / To:** Time span for the logical model date that has been used for the last run.

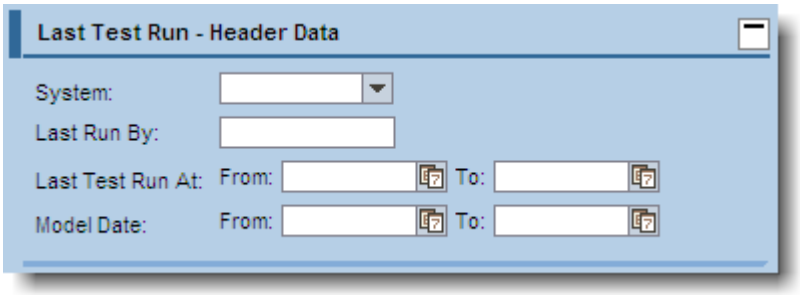


Fig. 4-15: Filtering for header data of the last run.

If you open the area **Last Run – Result**, see Fig. 4-16, you may filter for the result of the last test run. Here, you may specify the result of the last test run separately for each area of the test case (messages, configuration, BOM, and prices).

Please note that individual test cases may be run independently from test suites. Hence, it is possible that the result of the last test run of a test suite is green whereas at the same time the suite contains individual test cases the last result of which is red. This situation occurs if the test case has been green when the test suite has been run and has been executed afterwards with a red result.

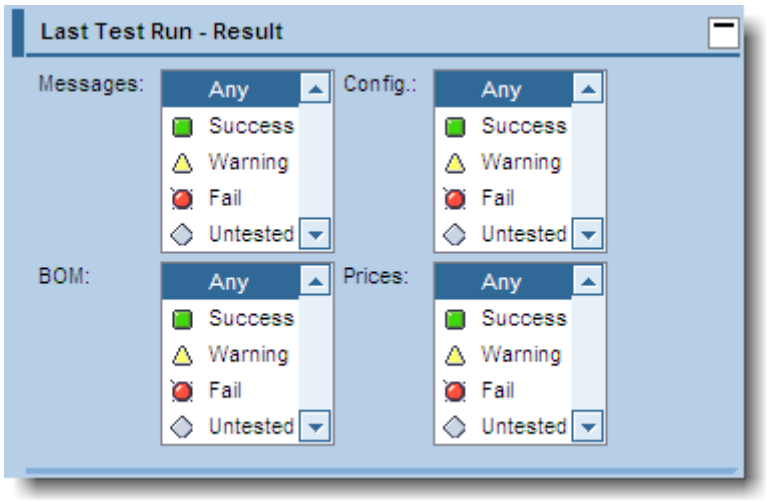


Fig. 4-16: Filtering for the result of the last test run.

If you open the area **Type**, see Fig. 4-17, you may filter test cases for the following criteria:

- **Type** of the test case: Here you may distinguish between complete and partial test cases. Read more about the differences between complete and partial test cases in Sec. 2.8.
- **Level:** This filter criterion is available in the load dialogue in the mass maintenance tab. Via this filter you may explicitly search for sub items of test cases. Read more about root items and sub items in Sec. 2.5.

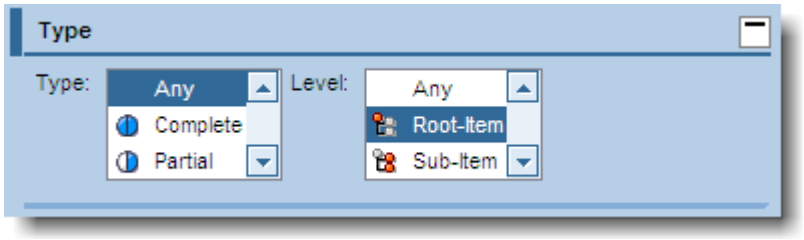


Fig. 4-17: Filtering for type and level.

If you open the area **Change History**, see Fig. 4-18, you may use the following filters:

- **Last Changed By**: User name of the user who did the last changes to the test case.
- **Created By**: User name of the user who created the test case.
- **Last Changed At: From / To**: Time span in which the test case has been had its last change.
- **Created At: From / To**: Time span in which the test has been created.

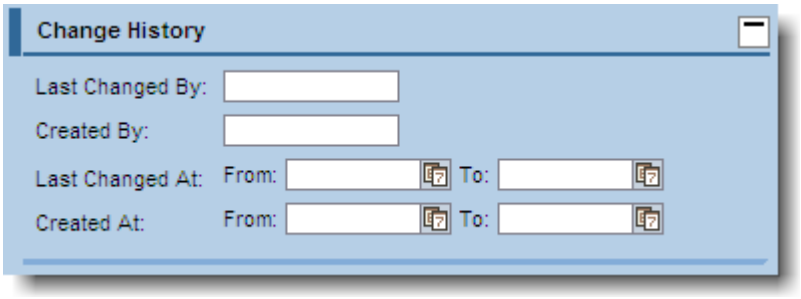


Fig. 4-18: Filtering for the change history of test cases.

4.5.2 Selection Table

In the selection area the following columns are available, see Fig. 4-10. The ordering of the columns reflects the ordering of the filter parameters, see Sec. 4.5.1. To adjust the visibility and / or the ordering of table columns, please refer to Sec. 4.1.2.

- First column group without heading
  - **Name**: Name of the test case
  - **Description**: Description of the test case
  - **KMat**: KMat of the test case
- Column group: **Bill of Materials – Header Data**
  - **Quantity**: Quantity of the test case for BOM explosion
  - **Plant**: Plant of the test case for BOM explosion
  - **Usage**: BOM usage
  - **Application** BOM application
- Column group: **Prices – Header Data**
  - **Calculation Schema**: Calculation schema used to test condition records
- Column group: **Last Test Run – Header Data**
  - **System**: Target system used for the last run
  - **Type**: Icon for the type of system: VC or IPC
  - **Last Run By**: User name of the user who triggered the last run
  - **Date and Time**: Time stamp of the last run
  - **Model Date**: Logical model date with which the run has been executed.

  Please note that individual test cases may be run independently from test suites. Hence, it is possible that the result of the last test run of a test suite is green whereas at the same time the suite contains individual test cases the last result of which is red. This situation occurs if the test case has been green when the test suite has been run and has been executed afterwards with a red result.
- Column group: **Last Run – Result**
  - **M**: Icon for the result of the last run in the messages area
  - **C**: Icon for the result of the last run in the configuration area
  - **B**: Icon for the result of the last run in the BOM area

- **P**: Icon for the result of the last run in the prices area

- Column group: **Type**
  - **Type**: Icons for partial or complete, see Sec. 2.8.
  - **Level**: Icons for root item or sub item, see Sec. 2.5.
- Column group: **Change History**
  - **Last Changed By**: User name of the user who did the most recent change
  - **Created By**: User name of the user who created the test case
  - **Last Change At**: Time stamp of the most recent change
  - **Created At**: Time stamp of the creation of the test case
- Column group: **Validity**
  - **From**: Start of the validity time span
  - **To**: End of the validity time span

4.6 Welcome

The welcome tab gives general hints for the application (Fig. 4-19). It contains hyperlinks to the following information sources:

- Sybit Model Tester customer area on the web.
- Sybit Model Tester support email address.
- Sybit Model Tester homepage

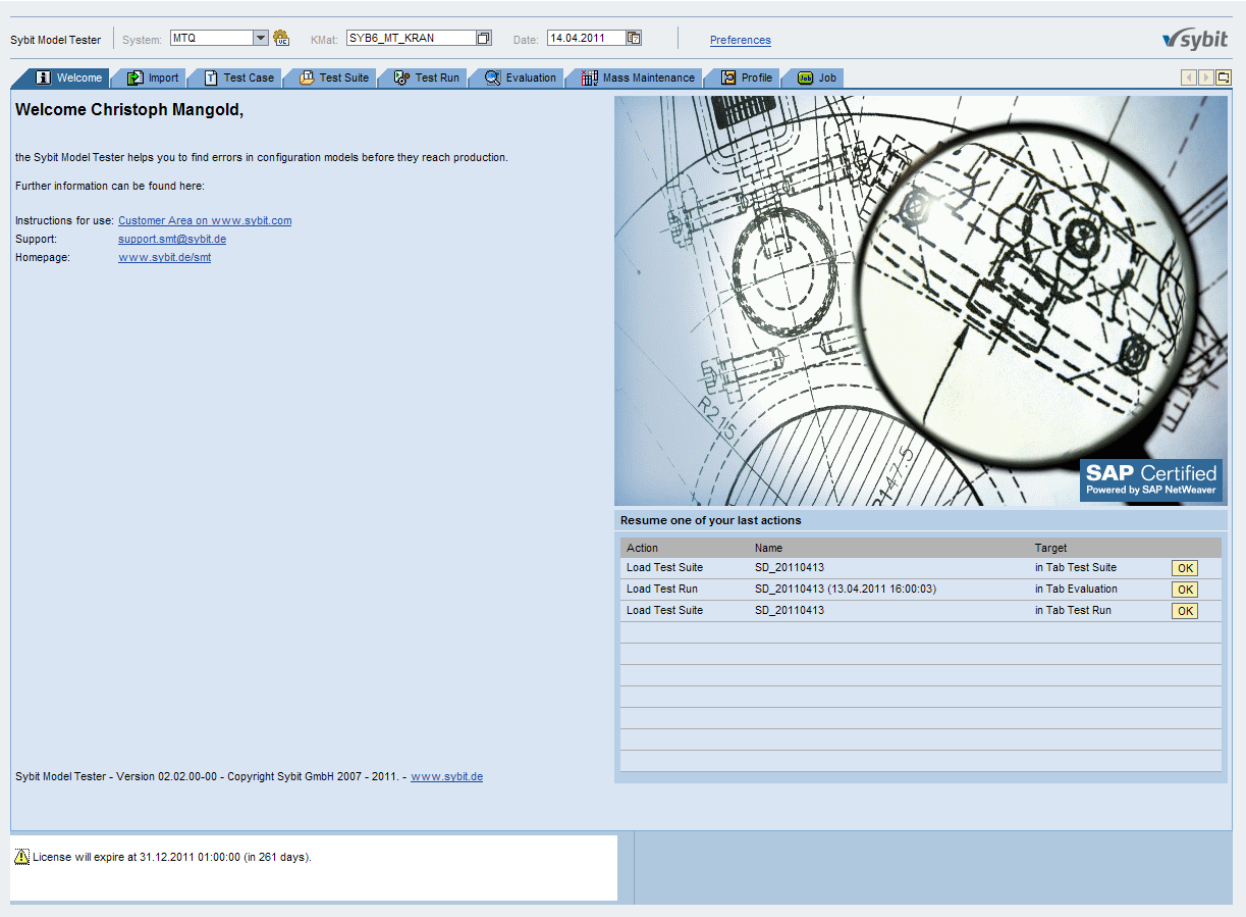


Fig. 4-19: Welcome tab

Additionally, the welcome tab enables you to resume your previous work. The table below the image contains your personal history where your most recent actions are listed at the top.

Each table line offers you the possibility to jump into a specific tab and load the object you where working on, at the same time. Here, the displayed actions depend on the KMat that is specified in the header area, see Section 4.2.1. Example: you will only see test cases in the table that test the KMat specified in the application header.

4.7 Import

The import tab contains two tools for automatic import of test cases. Each tool resides in a sub-tab of its own. In the SD tab you can import sales documents, create test cases from them and save them in the Model Tester. In the XML tab you can import test cases from external sources via predefined XML formats.

4.7.1 Sales Documents (SD)

In this tab you can import test cases from sales documents, see Fig. 4-20. To realize this, the Model Tester accesses the configuration data that is stored inside the document. User input values from the sales document are set as input data of the test case. User input values and system values are set as the configuration part of the expected result. The bill of materials of the test case is generated from the bill of materials that has been exploded inside the sales document.

In version 2.1, the layout of the import SD tab has been adapted to the layout of the common search component for test cases and test suites, see Section 4.5.1.

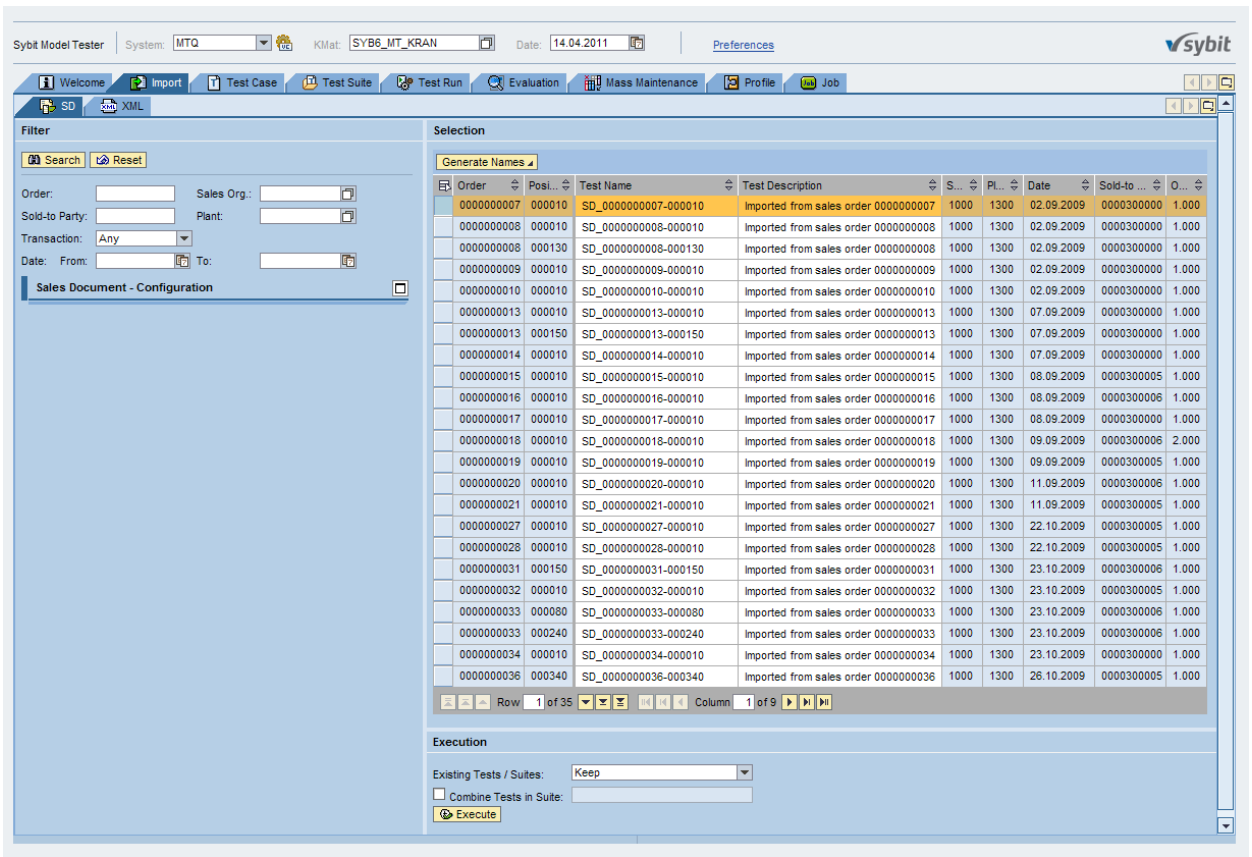


Fig. 4-20: Tab for import of sales documents

The following controls are available:

In the **Filter area** you can define filter criteria for the sales documents to display. There are filters for the following criteria:

- **Order**: The order number.

- **Sales organization**: The sales organization of the order.
- **Sold-to party**: The sold-to party of the order.
- **Plant**: The plant used in the order.
- **Transaction**: Sales order / inquiry / quotation / any.
- **Date: From / To**: Time span in which the order has been created.
- **Sales Document – Configuration**: In this table you may filter sales documents according to the characteristic values inside their configurations. This functionality is similar to the filter described in Section 4.5.1. In case of multi level configuration you may only filter for characteristic values on the root level.
- **Search**: The search button applies the filter criteria from below to all sales documents that are stored in the system. The result is displayed in the selection table on the right. If the system that is displayed in the header area is an IPC system, you will get the documents from the according ERP system, see Section 2.9.
- **Reset**: Deletes all filter entries made below.

In the **Selection** table all documents are displayed that match the specified filter criteria. If there are multiple KMats in one sales document and they all match the criteria, each KMat is displayed in a line of its own. To import a number of sales documents, just select the according lines in the table. Multi select is available. In the columns for test name and test description you may enter the name and the description of the test case that is generated from the respective line.

- **Generate Names**: With this button, you may generate test case names and descriptions automatically. The following options are available:
  - **New Names**: Generates unique names.
  - **Standard Names**: Generates test case names that may or may not already exist in the system. If you imported a sales document before and want to import it again – and overwrite the existing test case – this is the option you should choose.

In the **Execution** area you may trigger the test case generation.

- **Existing Tests / Suites**: Here you can specify how the import deals with test cases and test suites that are already present in the system. You may chose among:
  - **Keep**: Doesn't touch existing objects. If there is an object in the system that has the same name, the import will be canceled with an error message.
  - **Adapt**: Existing test cases / suites are adapted, i.e. overwritten, with the imported information.
  - **Delete**: Existing test cases / suites are deleted first. Subsequently, the import creates new objects in the system. If there is a test suite in the system that contains more test cases than you import here, those test cases will no longer be part of the test suite, once you have chosen the delete option. If you chose adapt, however, they are still part of the suite.
- **Combine Tests in Suite**: If you check this check-box, all imported tests are combined into a test suite with the specified name. You can overwrite the system-generated name manually.
- **Execute**: The execute button generates a test case for each selected table line and generates a test suite if required. Please note that generated test case names that where unique

at their time of generation may be not unique any more at a later time. If naming conflicts occur, the import is cancelled with an according error message.

4.7.2 XML

In this tab you can import XML-formatted test cases into the Model Tester, see Fig. 4-21... You can choose between two different XML formats:

- The SMT XML format directly reflects the Model Tester internal data structure for test cases.
- The Excel XML format is targeted towards an easy to use XML export from MS Excel. Accordingly, this format should be used if you want to create test cases in Microsoft Excel, instead of directly inside the Model Tester.

The specification of both formats is available as .XSD file. Please contact the Sybit Model Tester support, see Appendix B.

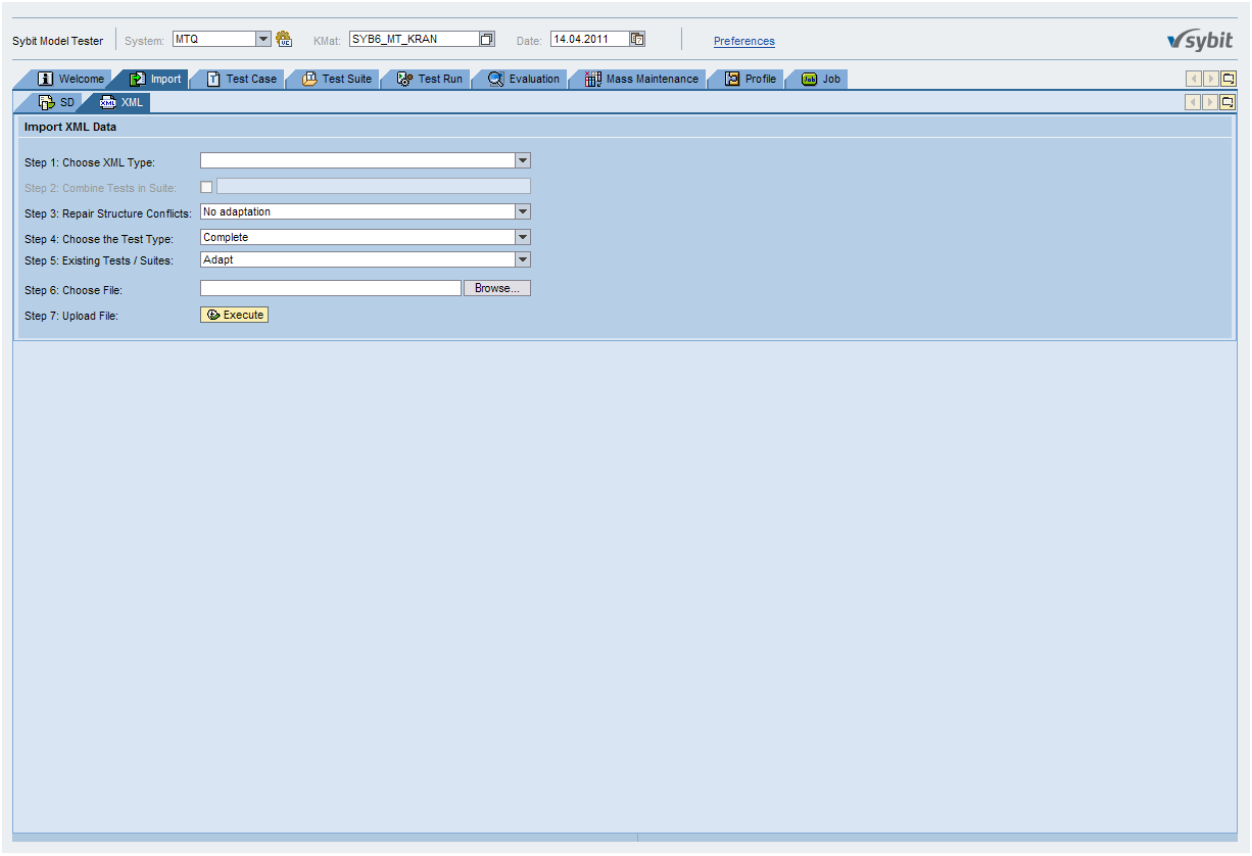


Fig. 4-21: Tab for Import of XML files.

The following controls are available:

- **Chose XML type:** Here, you specify if the import file complies to the SMT-XML or with the Excel-XML format.
- **Combine tests is suite:** This option is only available for Excel-XML. Here you may optionally define the name of the test suite in which all imported test cases should be contained.
- **Repair structure conflicts:** You may chose among the following options:

- **No adaption:** Test cases are imported exactly as specified in the XML file.
- **Level 1:** Sorts the characteristics in input and expected configuration according to the class order of the references configuration model (KMat).
- **Level 2:** Level 1 and additionally: Invalid position numbers – i.e. non matching sub-positions – of a multi level KMat are reassigned as long as they can be identified unambiguously by their KMat.
- **Level 3:** Level 1 and additionally: Invalid position numbers – i.e. non matching sub-positions – of a multi level KMat are reassigned. If they can not be identified unambiguously by their KMat, they are reassigned based on their order. Additionally, non-existing sub-positions are created and obsolete sub positions are deleted.

- **Choose test type:** Here you can specify whether the imported test cases should be partial or complete, see Section 2.8.
- **Existing tests / suites:** This option tells the system how to deal with naming conflicts, i.e. if objects with identical names already exist in the system. Available options are the same as in the import of sales documents, see Section 4.7.1.
- **Choose file**: Please pick the XML file to import. The file can be uploaded from your local computer or from a shared network drive.
- **Upload file:** Executes the import.

4.8 Test Case

The Test Case tab is used to create and maintain test cases, see Fig. 4-22. The tool is divided in four areas:

- Toolbar with buttons and fields for general test case information (at the top).
- Navigation table for multilevel test cases (left).
- Area for input configuration (middle).
- Area for the expected result (right).

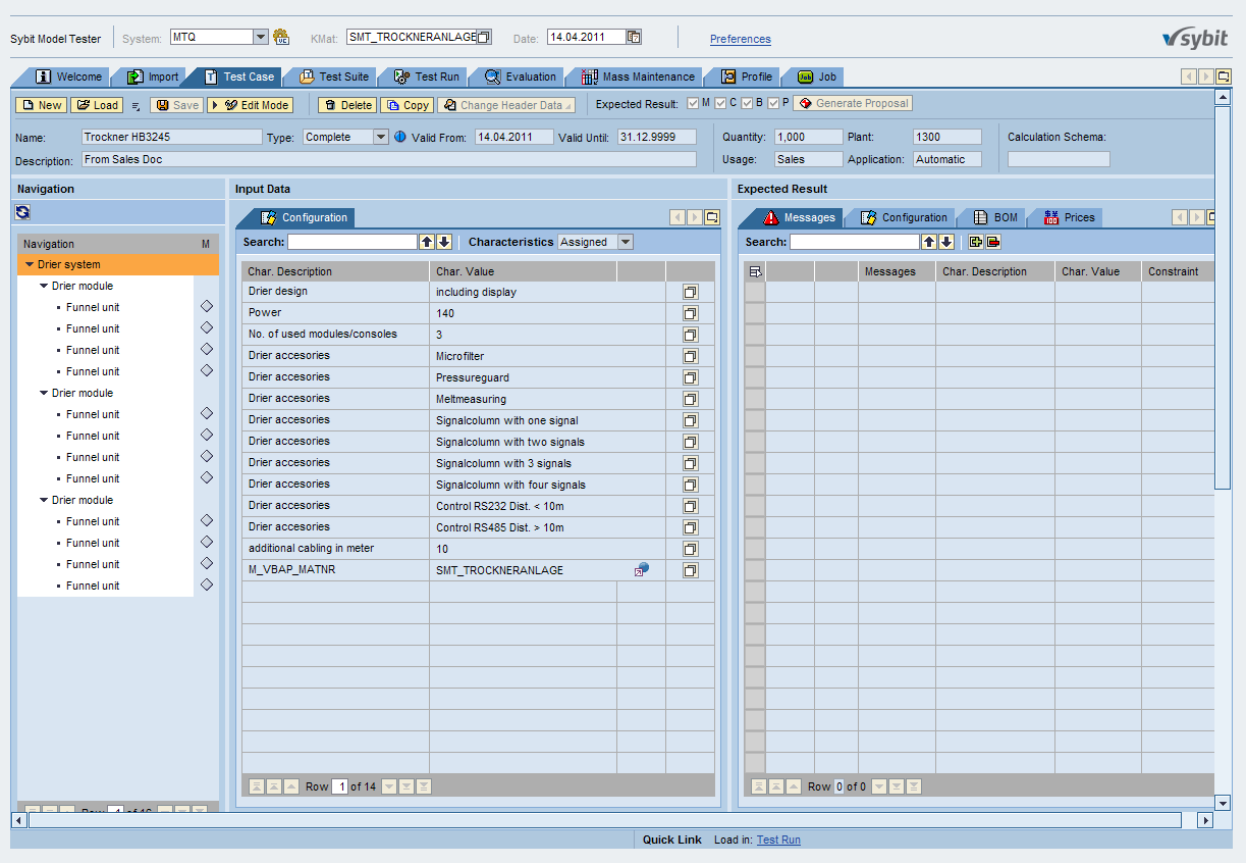


Fig. 4-22: Test case tab: Toolbar, navigation tree, input configuration and expected result.

4.8.1 Toolbar

The toolbar contains the following controls:

- **New**: Creates a new test case. Please notice that you can not create test cases if the system in the header is an IPC system, see Section 4.2.1.
- **Load**: Loads an existing test case. The load dialog contains the above described component for test case selection, see Section 4.5.
  Besides the load button, **the history drop down menu** enables you to quickly load test

cases from your personal history, see Section 4.4. The objects displayed here depend on the KMat that is specified in the header.

- **Save**: Saves the test case. The name of the test case is not permitted to be empty on save. If you change the test case name before saving the test case, you have then changed the name. This action does not copy the test case.
- **Edit Mode**: Switches from read-only to edit-mode and back. Please note that you may not switch into edit mode if you have an IPC system in the header, see Section 4.2.1.
  If you switch off the edit mode and have unsaved changes a popup will appear that asks if you want to save the changes, discard the changes, or cancel your action.
  If you switch on edit mode for a multi-level test case but your test case structure does not reflect the configuration structure of the KMat, a popup will appear where you can choose between the following options, see Fig. 4-23 (read more about test cases for multi level KMats in Section 2.5):
  - **Adapt**: The structure of the test case is adapted to the current structure of the KMat. Obsolete sub positions are deleted, new sub positions are inserted.
  - **Copy**: The test case is copied and the copy is loaded.
  - **Repair**: This option adapts the test case structure to the current structure of the KMat. You should use this option if position numbers of sub positions have been changed in the bill of materials which requires a re-assignment of sub items. The following actions are executed:
    - **Re-assignment**: The system finds the optimal assignment of sub items of the test case to sub positions in the current KMat.
    - **Deletion of obsolete sub items**: Sub items of the test case that can not be matched are deleted.
    - **Insertion of missing sub items**: Sub items that are missing are created automatically.
- **Delete** (available only in read-only mode): Deletes the loaded test case.
- **Copy** (available only in read-only mode): Creates a copy of the loaded test case and loads the copy.
- **Change header date:** This control contains the following buttons to change header data of the loaded test case.
  - **Change BOM:** Changes plant, application, and usage for the explosion of the bill of materials.
  - **Change Calc. Schema:** Changes the calculation schema of the test case.
  - **Change Validity:** changes the validity time span of the test case.
- **Generate Proposal:** This button helps you to fill the expected result directly from the VC model: It passes the input configuration to the VC and loads the generated result into the checked areas of the expected result. The check boxes labeled with M, C, B, and P represent the areas for messages, configuration, BOM, and pricing, respectively.
  A click on the generate proposal button automatically refreshes the navigation tree on the left. If you don't like this behavior, you can easily switch it off as described in Section 4.1 in the chapter of test case parameters. For performance reasons, sub positions that have just been created do not contain an expected result. To get an expected result for them, just press generate proposal again.
  If clicking on generate proposal triggers the creation of new sub items in the test case, the

expected result of those sub items is not filled automatically due to performance reasons. I.e. the expected result of newly created sub positions is always empty. To visualize this, the navigation tree shows a grey icon in those cases.
To give you an immediate feedback of error and warning messages in your test case, the Model Tester displays icons inside the navigation tree on the left. The icons have the following meaning:

- **Red light:** The test case contains an error message in the expected result of this hierarchy level.
- **Yellow light:** The test case contains a warning message in the expected result of this hierarchy level.
- **No icon:** The expected result contains no messages in this hierarchy level.
- **Grey icon:** This hierarchy level has just been created by clicking on generate proposal. Hence, the expected result on this level is still empty. To fill it, click generate proposal again.

- **Name**: Name of the test case. Empty names are not permitted. Test case names must be unique per KMat. Tests with the same name may exist for different KMats.
- **Description**: Description of the test case.
- **Type**: Type of test, see Section 2.8. You may choose complete or partial. If in doubt always use complete which is also the default.
- **BOM Plant**: Plant for BOM explosion.
- **BOM Usage**: Usage for BOM explosion (sales / engineering).
- **BOM Application**: Application for the BOM, e.g. SD01 for Sales, PP01 for production, etc if it can not be determined automatically from the configuration profile of the KMat.
- **Valid from / Valid until:** Defines the validity period for the test case. The fields can not be edited directly but only by means of the change validity button.
- **Quantity**: Here you may adjust the quantity of the configured instance in your test case. Most often you will have a quantity of 1.0. This option is mainly relevant for test cases that have been generated from sales documents.
- **Calc. Schema**: If you want to test prices with your test case, you need to define the appropriate calculation scheme here. If your test case is not about prices, leave the field empty. In the test run, this will result in a grey icon.

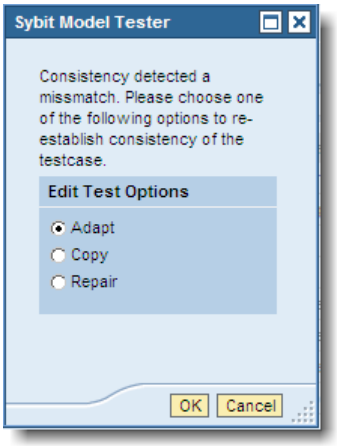


Fig. 4-23: Popup when changing into edit mode with a multi-level test case that has an invalid structure.

4.8.2 Navigation Area

The navigation area is used to change configuration structure levels in test cases for multi-level KMats. In case of multi-level configuration the input configuration and the expected result is displayed and edited for each level separately. To navigate to a different level, just click on the name of the according KMat in the table.

- **Refresh: This button triggers the synchronization of the test case structure with the current** configuration instance inside the LO-VC. If you enter a characteristic value in the input table that provokes, for example, a new sub-KMat then you must press the refresh button to display the updated test case structure. A click on refresh triggers a BOM explosion in the back-end that is based on the current input configuration. The result of the BOM explosion determines the new test case structure.

The icons in the navigation tree have the following meaning:

- **Red light**: The test case contains an error message in the expected result of this hierarchy level.
- **Yellow light**: The test case contains a warning message in the expected result of this hierarchy level.
- **No icon**: The expected result contains no messages in this hierarchy level.
- **Grey icon**: This hierarchy level has just been created by a click on generate proposal. The expected result on this level is empty. To fill it, click generate proposal again.

4.8.3 Input Data

In the input data section you specify the part of the test case that you would type into CU50 if you were in a manual testing scenario (Fig. 4-22).

- **Search**: You can use this input field to search for characteristic names and values. Arrow up / arrow down searches up / down, respectively. The search engine always scans both, the language dependent and the language independent descriptions.
- **Any / Assigned**: Here you may choose between two different view modes in the configuration table. If you choose **assigned** (which is the default after loading a test case) you only see characteristics that have a value. If you switch to **any** (which you can do in edit-on mode only) you see all characteristics, i.e. also those that have no values. In the **any** view,

the application makes sure that characteristics from input and expected result are always aligned in the table.
Switching between assigned and any always affects input and expected configuration.

- **Char. Value column**: Direct input of characteristic values. If you chose the option **both** in the presentation section in the preferences popup then you will see two columns for the characteristic name and two columns for the value. The left column always contains the language independent, the right column contains the language dependent description, respectively.
  Table 4-2 shows the content of language independent and language dependent formatting. You may input your values in either format.
  In the test case database, the Sybit Model Tester stores language independent values only. This ensures that test cases may be used across languages.

Value Type	Language Independent	Language Dependent
Date Value	TTMMYYYY e.g.: Dec. 15th, 2009: 15121009	MM.TT.YYYY e.g.: Dec. 15th, 2009: 12.15.2009 (this format depends on user specific preferences, see transaction SU01)
Numeric Value	No mask and no unit, decimal point is used as decimal separator e.g.: 12345.67	Formatted with mask and unit e.g.: 12,345.67 pc
Currency Value	Unformatted without currency e.g.: 12345.67	Formatted with currency e.g.: 12,345.67 EUR
Character Value	Language independent identifier e.g.: CMF_PLUG	Language dependent description e.g.: Comfort plug
Time Value	SSMMss e.g.: 15 hours 26 min 57 sec: 152657	SS:MM:ss e.g.: 15 hours 26 min 57 sec: 15:26:57

Table 4-2: Mapping of language independent and language dependent characteristic values.

Please note that the current version ot the Sybit Model Tester only supports interval values in the format value1 – value2 (from value1 upto value2). The following formats are not supported:

- > value1 (greater than value1)
- >= value1 (greater or equal than value1)
- < value1 (less than value1)
- <= value1 (less or equal than value1)
- >value1 - <value2 (greater than value1 and less than value2)
- value1 - <value2 (greater or equal than value1 and less than value2)
- >value1 – value2 (greater than value1 and less or equal than value2)

- **Type column**: In this column – that has no column header – the application informs you about special types of characteristic values. The following two icons may be displayed:

- **Object characteristic**: All values that are assigned to reading object characteristics carry the icon for object characteristics. Reading object characteristics are used to transport values e.g. from the sales document into the configuration.
  The application marks these characteristics because in a test run, object characteristic values are set in a separate execution steps before anything else is set.
- **Delete master data default:** In some cases you may not only wish to set values in the input configuration, but also to delete certain default values. This applies particularly, if you want to create a configuration that requires that you delete a value that is set as a master data default. To learn how to mark a value as a deleted master data default, see below in the description of the value help popup for characteristic values.

- **Value help button**: The value help button opens a popup dialogue that helps you to find characteristic values.

A click on the value help button opens a value help popup window, see Fig. 4-24. In the left **Select** column you may choose predefined characteristic values. If the characteristic is extensible, you may add additional, non-predefined values in the last line of the table. Here, each input of a new value needs to be confirmed by pressing the Enter key. Then, the value is moved into the upper part of the table.

The check boxes of those values are active, that are marked as master data default in the column **Del MD Def**. If you check the check box of a value here, the value will be marked with the appropriate icon in the input configuration (see above). If a value is marked as a deletion of a master data default this has the same effect as if – e.g. in the transaction CU50 – you delete a system set default value manually.

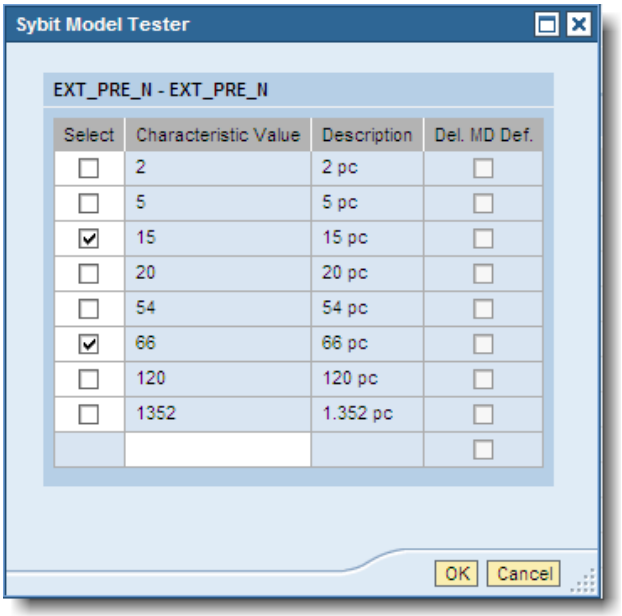


Fig. 4-24: Value help window for characteristic values.

4.8.4 Expected Result

The expected result contains the following tabs. One for expected messages, one for the expected configuration, one for the expected bill of materials, and one for expected prices, respectively.

4.8.5 Expected Messages

In the tab for **expected messages** (Fig. 4-22, on the right) you can specify messages for the conflicts and warnings that you expect the LO-VC to generate upon your input configuration. E.g., if you want to test if a characteristic is marked required, you just leave it empty in the input. If the expected messages contains the warning CSTIC_INCOMPLETE, then the test run succeeds if (and only if) the characteristic is required.
With the search input field you can conduct a full text search on the table. You may choose among all possible messages by means of the drop down box. The messages are listed in Appendix A.

If a message can be assigned unambiguously to a characteristic, you may jump directly to the characteristic in the input configuration. To do this, just click on the row selector of the message. The row selector of a row is the button at the beginning of each line that enables you to select the row. If you click here, the input configuration table will scroll so that the according characteristic is displayed in the first line.

4.8.6 Expected Configuration

In the tab for the **expected configuration** (Fig. 4-25) you specify the characteristic values that you expect the system to set based on your input configuration. For your convenience, the values you set in your input configuration are copied automatically into your expected result. Toggling between complete view and quick view also affects the input configuration. For a description of these options, see Section 4.8.3.

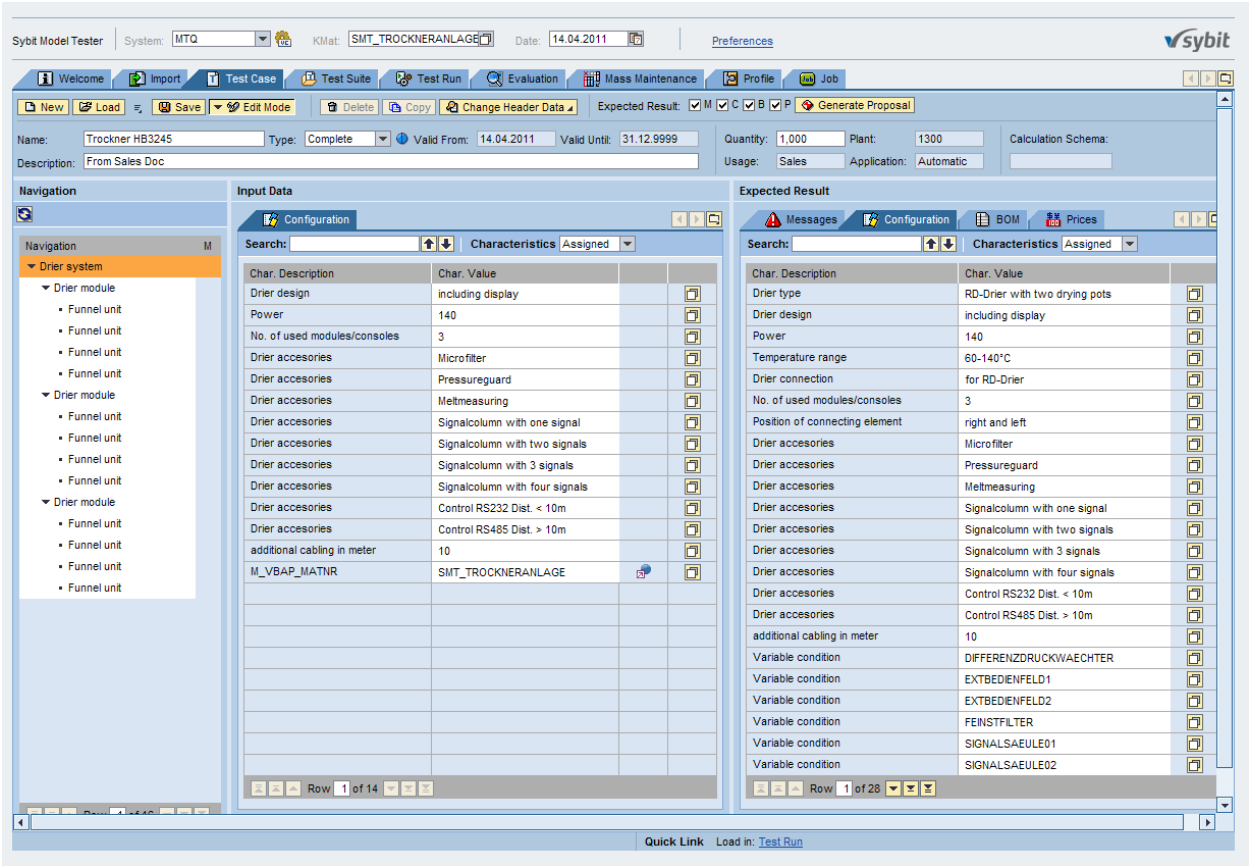


Fig. 4-25: Test case tab: Configuration area of the expected result.

4.8.7 Expected Bill of Materials (BOM)

The tab for the **expected bill of materials (BOM)** contains the result of the BOM explosion (Fig. 4-26). Again, you can use the search field to conduct full text search on the table.

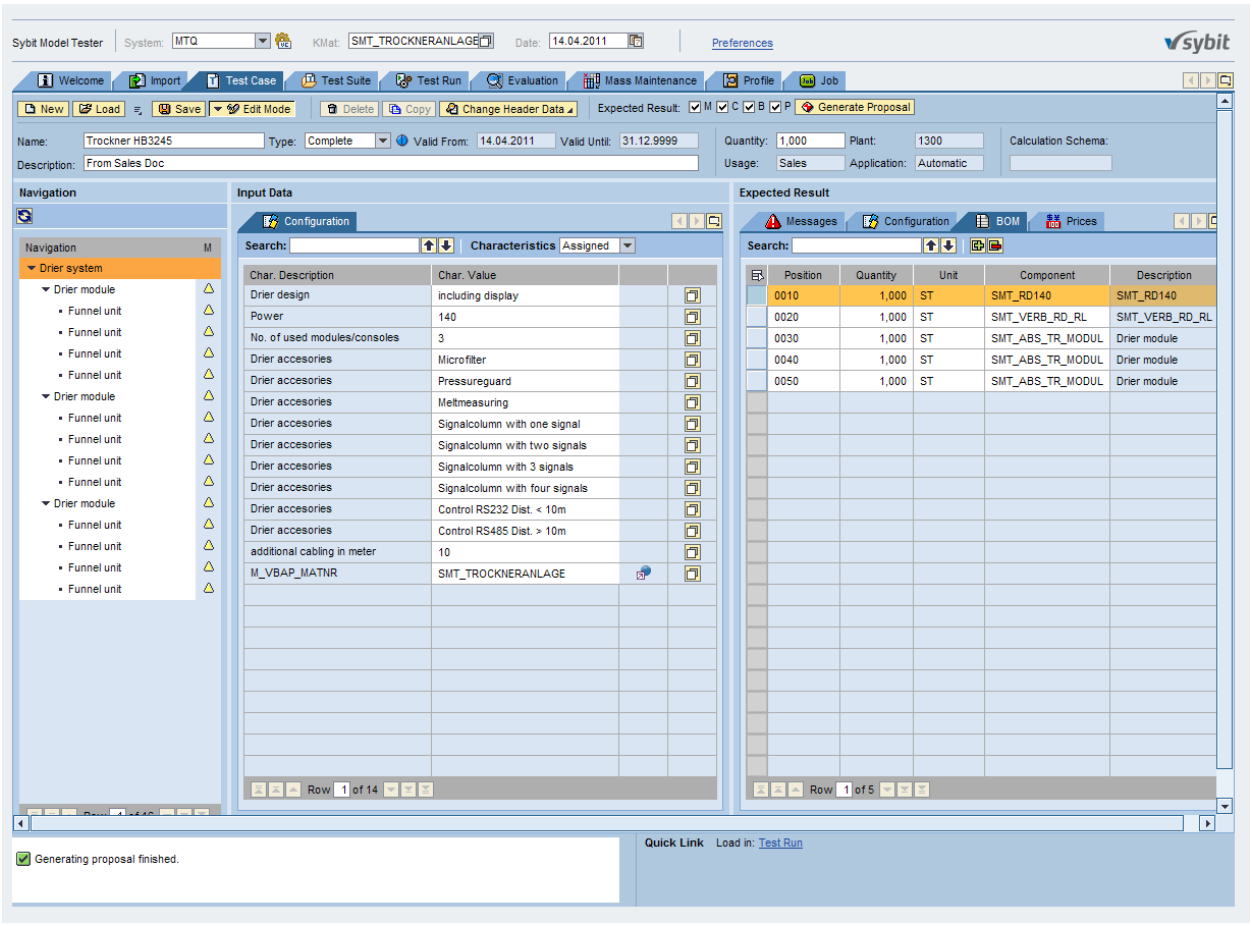


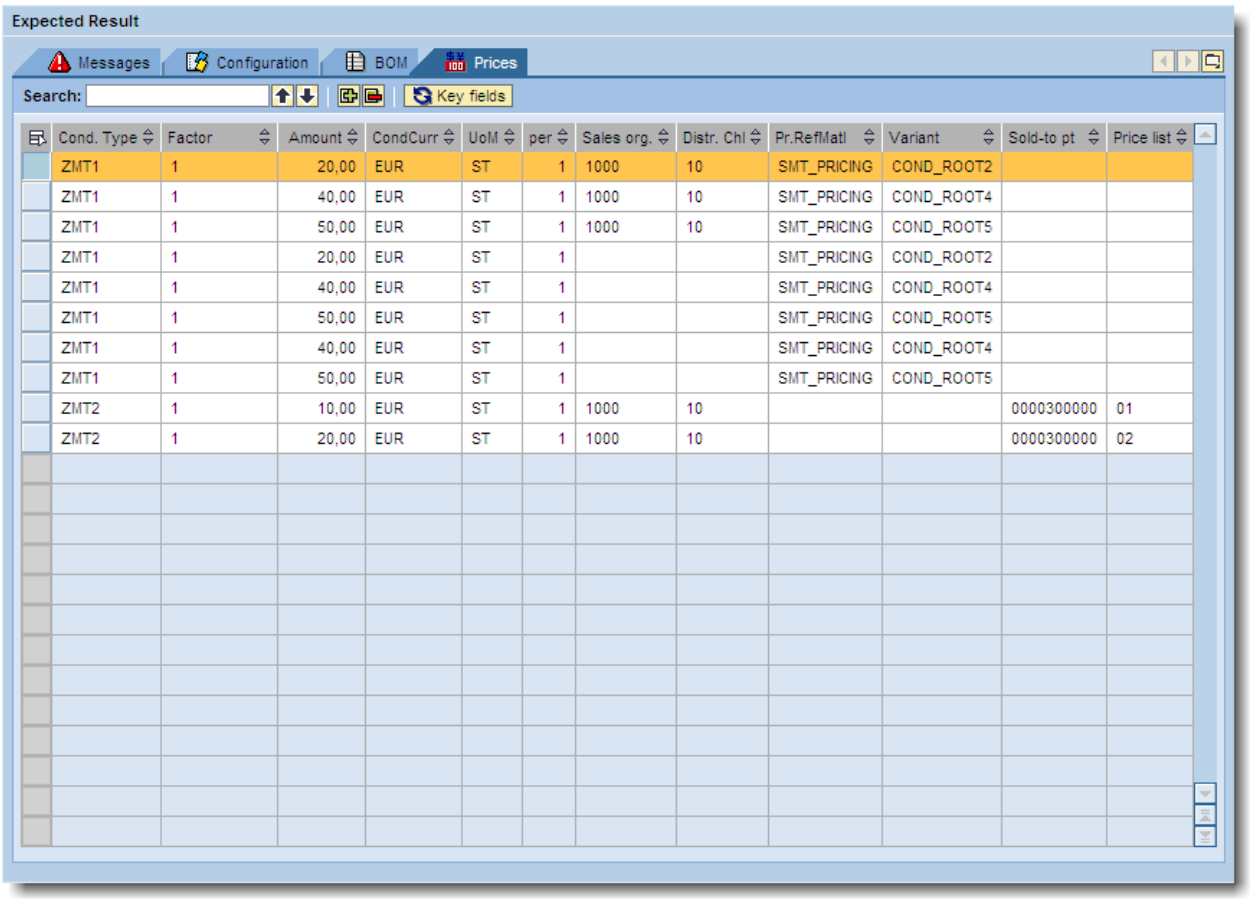
Fig. 4-26: Test case tab: BOM area of the expected result.

4.8.8 Expected Prices

In the tab for expected prices you specify all condition records relevant for the variant conditions of your configuration, see Section 2.3. Only those condition records are tested that are relevant to the calculation schema specified in the header of the test case.

This list of condition records is displayed in one single table in the expected result and may also be edited there. The table columns are as follows (see Fig. 4-27):

- **Condition type**: Name of the condition type.
- **Factor**: During model evaluation, the VC calculates a factor for each condition record. This factor is displayed here.
- **Amount, CondCurr, UoM, per:** Static columns for condition records see above.
- **Dynamic columns:** Since each access may define its own key columns, we see all key columns of all accesses here. Columns that are not needed in an access stay empty for all condition records of this access.



Cond. Type	Factor	Amount	CondCurr	UoM	per	Sales org.	Distr. Chl	Pr.RefMatl	Variant	Sold-to pt	Price list
ZMT1	1	20,00	EUR	ST	1	1000	10	SMT_PRICING	COND_ROOT2		
ZMT1	1	40,00	EUR	ST	1	1000	10	SMT_PRICING	COND_ROOT4		
ZMT1	1	50,00	EUR	ST	1	1000	10	SMT_PRICING	COND_ROOT5		
ZMT1	1	20,00	EUR	ST	1			SMT_PRICING	COND_ROOT2		
ZMT1	1	40,00	EUR	ST	1			SMT_PRICING	COND_ROOT4		
ZMT1	1	50,00	EUR	ST	1			SMT_PRICING	COND_ROOT5		
ZMT1	1	40,00	EUR	ST	1			SMT_PRICING	COND_ROOT4		
ZMT1	1	50,00	EUR	ST	1			SMT_PRICING	COND_ROOT5		
ZMT2	1	10,00	EUR	ST	1	1000	10			0000300000	01
ZMT2	1	20,00	EUR	ST	1	1000	10			0000300000	02

Fig. 4-27: Test case tab: Prices area of the expected result

The following controls are available:

- **Add row / delete row** : For manual maintenance of pricing information, you may use these buttons to add / delete rows in the table.
- **Refresh key fields**: This button analyses the calculation schema specified in the header of the test case and determines all key columns of all relevant accesses. New columns are added as empty columns in the table. Columns that cease to exist are deleted.

> **Caution**: If columns are deleted, all rows are deleted that have a non-empty value in the deleted column.

The Sybit Model Tester always tests if the following two rules are met, independently of the table in the expected prices tab

- **At least once:** Each variant condition must occur in a least one condition record.
- **Last access:** If a variant condition is contained in a condition record of an access of an access sequence, then it must also be contained in the last access of the same access sequence. This rule makes sense as access sequences go from specific to general.

If one of those rules is violated, an according error message occurs in the message tab, see Appendix A.3.

4.9 Test Suite

Test suites are used to organize test cases and help you to execute, evaluate, maintain, and re-find them. The test suite tab contains the central tool to view and edit test suites (Fig. 4-28).

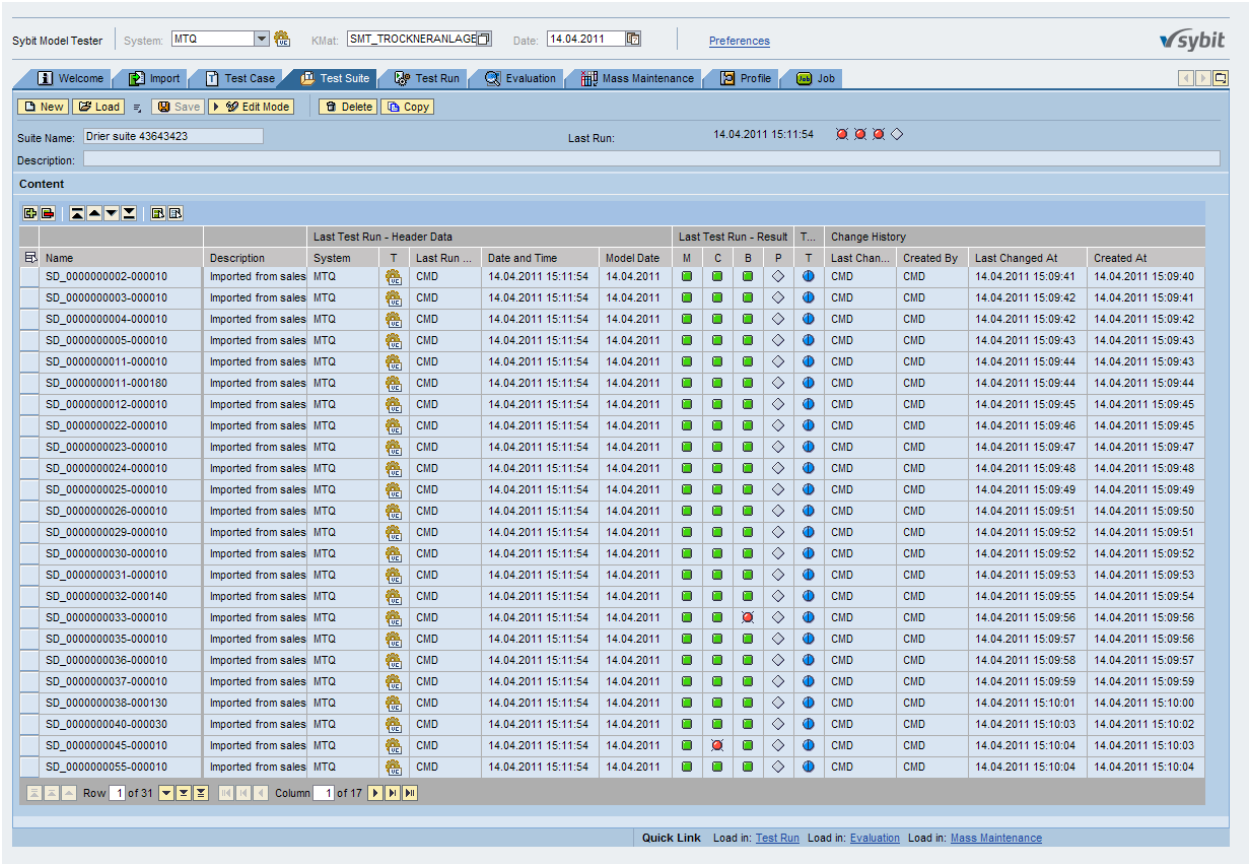


Fig. 4-28: Test suite tab.

The following controls are available:

- **Toolbar**: The buttons **New / Load / Save / Edit Mode / Delete / Copy** fulfill exactly the same functions as discussed in the above section on the test case tab.
- **Name / Description / Last Run:** These fields give general information about the suite. The rules for test suite names match those for test case names, see Section 4.8.1. The last run fields denote the date and time when the test suite was last executed.
- **Insert :** Opens a popup window with the common selection component for test cases and test suites, see Section 4.5. Here you may search for test cases and add them to the end of the test suite you are currently editing.
- **Remove :** Removes the selected test cases from the suite. The test cases are not removed from the system.
- **Move Up / Move Down** : The buttons move the selected test cases upwards / downwards within the suite.
- **Move to the Top / Move to the Bottom**: Moves the selected test cases to the very top / bottom of the list.

- **Select All / Deselect All**: Selects / deselects all test cases within the test suite.

4.10 Test Run

The test run tab contains the tool to execute test cases and suites (Fig. 4-29).

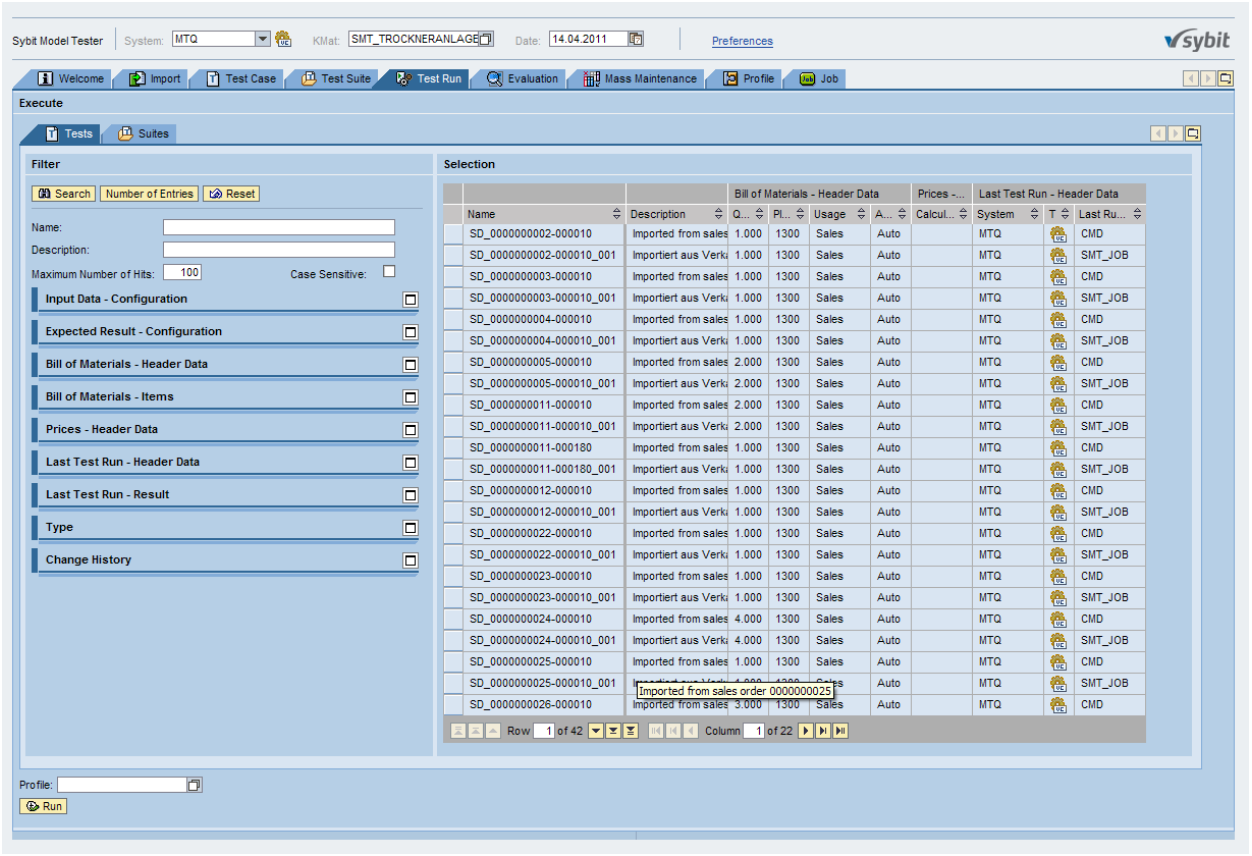


Fig. 4-29: Test run tab.

The following controls are available:

- **Selection Area**: The selection area contains the above described common interface component for test case and test suite selection, see Section 4.5. Before you run a test or a test suite, you must select it here.
- **Profile:** For each test run you may specify a profile that effects the evaluation of test cases contained by the run, see Section 4.13. In particular a profile may specify that certain errors are ignored on evaluation, i.e. they don't result in a red light. If you don't specify any profile, all information in the test case are considered relevant. Any error during evaluation will result in failure of the respecting test case.
- **Run button**: The test case / test suite selected above is executed in the system that is displayed in the header area.
  For the differences between test runs in the LO-VC and the IPC, please refer to Section 2.9.
  The execution date, i.e. the logical model date where the run is executed is taken from the date in the header.

4.11 Evaluation

The evaluation tab contains the tool to evaluate test runs, see Fig. 4-30.

The Sybit Model Tester application is designed to remember all test runs including the input, the expected result, and the factual result. Hence, for each test case and test suite all test runs can be evaluated at any time. That way, you can retrace how your configuration model behaved at the execution time of each test run.

SMT 2.2

There are two sub tabs where the information about test runs is displayed in two different manors:

- **By Test Case** (see Section 4.11.1): In this sub tab you see the results of a test run on a per test case basis. For each test case that has been executed in the run you see the result for each area.
- **By Error** (see Section 4.11.2): In this sub tab the focus is on the errors that occurred in the test run. For each individual error you get the number of test cases in which the error occurred and a list of all affected test cases.

In principle, both sub tabs contain the same information about the test run. From both sub tabs you may open the window for the detail analysis of single test cases, see Section 4.11.7.

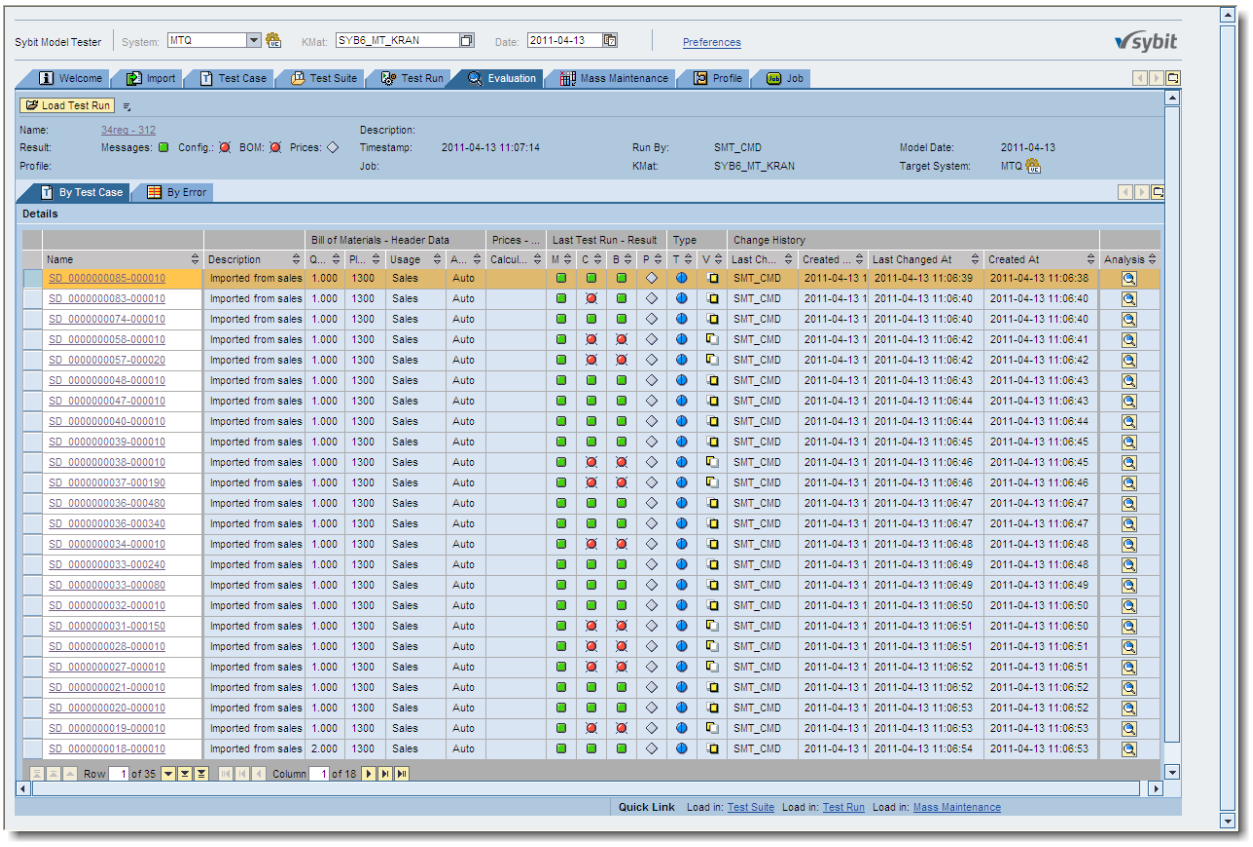


Fig. 4-30: Evaluation tab: By Test Case.

The following global controls are available:

- **Load Test Run**: Loads the run of a test case or a test suite for evaluation. The button opens a popup dialogue that works similar to the common selection component for test cases and test suites as described in Section 4.5.
- **Test run information area**: The upper area contains all relevant information on the test run under evaluation. The traffic light icons are yellow / red if at lease one test case have a yellow / red traffic light icon in the respective area.

4.11.1 Sub tab: By Test Case

- **Details table**: If you loaded a test suite, the details table contains the list of test cases within the suite. To give you a quick overview, each test case directly shows its state for each of its areas (messages, configuration, BOM, and prices). Only if all test cases in a test suite are green in one area, this area is also green for the test suite.
  If you loaded a test run with one test case only, the table contains one single line only.
  The columns in the table are similar to the columns in the common selection component for test cases, see Section 4.5. The following columns are available additionally:
- **Column Current Version (V)**: The icon in this column tells you if the test case in the respective line is the current version. If the test case has been changed since this run, it is not any more the current version.
- **Analyze** button: To analyze a test case in detail you may press the analyze button which opens the analysis popup window, see Section 4.11.7.

4.11.2 Sub tab: By Error

SMT 2.2

Compared with the By Test Case tab, this tab inverts the way information is displayed, see Fig. 4-31. In the By Test Case tab the focus is on test cases and for each test case you may analyze all errors contained. In contrast, in the By Error tab the focus is on the error. Likewise, for each error occurred you may display all test cases that contain the error.

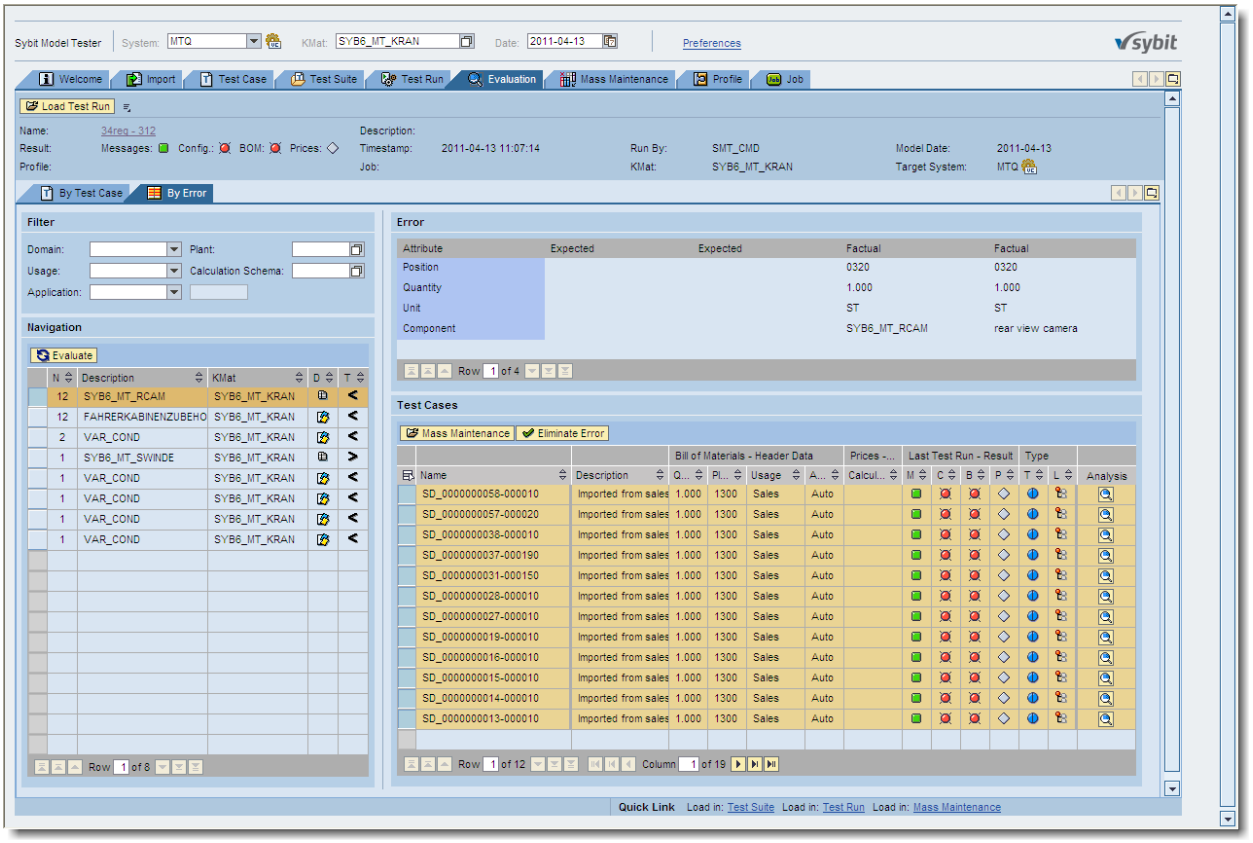


Fig. 4-31: Evaluation tab: By Error.

The By Error Tab contains four areas that are described in the following.

4.11.3 By Error Tab: Filter

SMT 2.2

In the filter area you may pre-select the test cases that you want to analyze. The smaller the set of filtered test cases, the faster you will get your evaluation results in the navigation area. The following filters are available:

- **Domain**: Restricts the set of evaluated test cases to those that contain errors in the specified area.
- **Plant**: Restricts the set of evaluated test cases to those that test the specified plant.
- **Usage**: Restricts the set of evaluated test cases to those that use the specified BOM usage.
- **Calculation Schema**: Restricts the set of evaluated test cases to those that use the specified calculation schema.
- **Application**: Restricts the set of evaluated test cases to those that use the specified BOM application.

4.11.4 By Error Tab: Navigation

SMT 2.2

If you click on the push button **Evaluate** the system does the following. First, the filter criteria specified in the filter area are applied to all test cases in the test run under evaluation. Then, all test cases that match the filter are analyzed to determine a list of all errors that occurred. This list is displayed in the navigation table. Each error is displayed in a line of its own.

The following columns are available:

- **Number**: This column contains the number of test cases that contain the specified error. Since one error may occur in a test case more than once, this number may exceed the number of overall test cases. Since errors that occur most often deserve your utmost attention the table is sorted by descending number.
- **Description**: This column contains a short description of the area. Depending on the area the following information is displayed:
  - Messages: Characteristic name if available, otherwise the message type.
  - Configuration: Characteristic name.
  - Bill of materials: Component name.
  - Prices: Variant condition if available, otherwise empty.
- **KMat**: The KMat of the test case. In case of multi level test cases, this column contains the KMat of the sub-item that contained the error, see Section 2.5.
- **Domain**: Icon for the area where the error occurred.
- **Type**: Icon for the type of error:
  - < - Not expected in the test case: The factual result contains an entry that is not contained by the expected result. The icon is supposed to display that the expected result (on the left) is smaller than the factual result (on the right).
  - > - Not found in the factual result: The expected result contains an entry that misses in the factual result. The icon is supposed to display that the expected result (on the left) is greater than the factual result (on the right).
  - ≠ - Not equal: Two entries have been found in the expected and the factual result that could be matched. However, the entries differ in at least one component.

If you select an error in the navigation table (by clicking on the selector), the system fills the areas Error and Test Cases on the right hand side of the By Error tab with detailed information concerning the selected error.

Via the global preference that switches between language dependent and language independent display of navigation structures (see Section 4.2.2) you may switch the language dependency of the columns KMat and Description.

4.11.5 By Error Tab: Error

SMT 2.2

This area displays the error selected in the navigation table in detail. In the table, the expected result is on the left hand side, the factual result is on the right.

Each line of the table contains one component of the erroneous entry.

Example: For an error in the configuration area the table contains two lines for characteristic name and characteristic value, respectively. However, for an error in the BOM the table contains four lines for position, quantity, unit, and component, respectively.

Via the global preference that switches between language dependent / language independent display for characteristics, see Section 4.2.2, you may also influence the columns of the error table.

4.11.6 By Error Tab: Test Cases

SMT 2.2

The table in the test cases area contains the list of all test cases that contain the error displayed in the error area. If you click on the button in the analyze column of the table, the analysis window will be displayed that lets you analyze the respective test case in detail, see Section 4.11.7.

Additionally, the area contains the following controls:

- Button **Mass Maintenance**: This button navigates directly into the mass maintenance tab (see Section 4.12) and loads the test cases that are selected in the table. In the mass maintenance tab, you may edit the test cases at will.
- Button **Eliminate Error**: This button eliminates the displayed error in all selected test cases. To this end, in each test case the respective aspect of the expected result is replaced with the according aspect of the factual result. For the different error types this means the following:
  - < - Not expected in the test case: The factual result is added into the expected result.
  - > - Not found in the factual result: The expected result is removed.
  - ≠ - Not equal: The expected result is replaced by the factual result.

  The eliminate error button has the same effect as the adapt result button in the analysis window, but it fixes one single error only.
  Please note: if the input configuration of a test case has been changed since the test run you are evaluating, the system prohibits error correction via the eliminate error button. Otherwise you would adapt the expected result of a test case according to a deprecated input configuration. Since these kind of changes can hardly be tracked, they inevitably lead to confusion.

4.11.7 Analysis Window

The analysis window is divided into an area for information and navigation on the left and the main analysis area on the right.

The upper area on the left contains the test run area which displays the name of the test case or test suite that has been executed in this test run. If the test run contains a test suite, you may navigate within the test suite using the buttons **Previous Test case** and **Next Test case**.

The test case area below contains a tool bar with the following button:

- **Adapt Result**: With a click on this button you overwrite the expected result of the current test case with the factual result determined by the test run. Please keep in mind that icons don't become green instantly. To achieve this, you have to re-run the test.
  For test runs in the IPC, the adapt result function is not available, see Section 2.9 and Section 4.2.1.
  If your test run has been executed with a profile, the factual result contains the displayed information only. Only this displayed information is copied to the expected result. If you run into this scenario, you will be informed by a popup dialogue that you must confirm.
  Please note that you may adapt the result only of those test cases that didn't change their input configuration since the test run displayed here.

In the toolbar below, an information area shows you important header data of the test case under analysis. Further down in the navigation area you can navigate test cases for multi-level KMats similar to the test case tab, see Section 4.8. For each KMat, the navigation tree shows the respective result divided up in messages (**M**), configuration (**C**), bill of materials (**B**), and prices (**P**).

To navigate inside the test case the following controls are available:

- **Previous Error**: If you analyze a multi-level test case, this will set the focus on the level with the previous error in the navigation tree.
- **Next Error**: If you analyze a multi-level test case, this will set the focus on the level with the next error in the navigation tree.

The following two display modes are available

- **Generalization**: Red icons are raised in the tree from the bottom up. If the configuration area of a lower level has a red icon, then all configuration areas of all superior levels will also have a red icon. However, the configuration area itself may have a green icon, which is then displayed in the tab on the right side.
- **Specialization**: Red icons are only shown at the level of the tree where the problem occurs. If your tree is not fully expanded and you only see green icons, the test case may have failed anyway which results in red icon in the summary.

The **analysis area** has tabs for messages, configuration, bill of materials, and prices, respectively. Each tab is split into two halves. On the left there is a table containing the expected result of the test case. The right side contains a table that shows the factual result. Lines that do not match between left and right are marked with bright red color.

4.11.8 Analysis Window: Messages Tab

Fig. 4-32 shows the messages tab. On the left of the analysis area you see that the expected result did contain no messages, whereas the factual result (right side) reports some messages.

You may choose among the following view modes via the drop down box:

- **Show conflicts only**: Only those lines are displayed where expected and factual results do not match. Those lines are red.
- **Show relevant messages** (default value): Rows and columns that are marked irrelevant are hidden. The following data is treated as irrelevant:
  - **Columns:** If you trigger a test run on a LO-VC system, all columns are relevant. If you trigger a test run on an IPC system, the constraint column is irrelevant. Read more about this in Section 2.9.
  - **Lines:** Messages are treated irrelevant if and only if they refer to a characteristic that is marked to be irrelevant by a profile.
- **Show all messages**: Irrelevant information is also shown and marked with blue background. This affects both, the irrelevant constraint column in IPC test runs and the irrelevant messages that affect characteristics marked irrelevant by a profile.

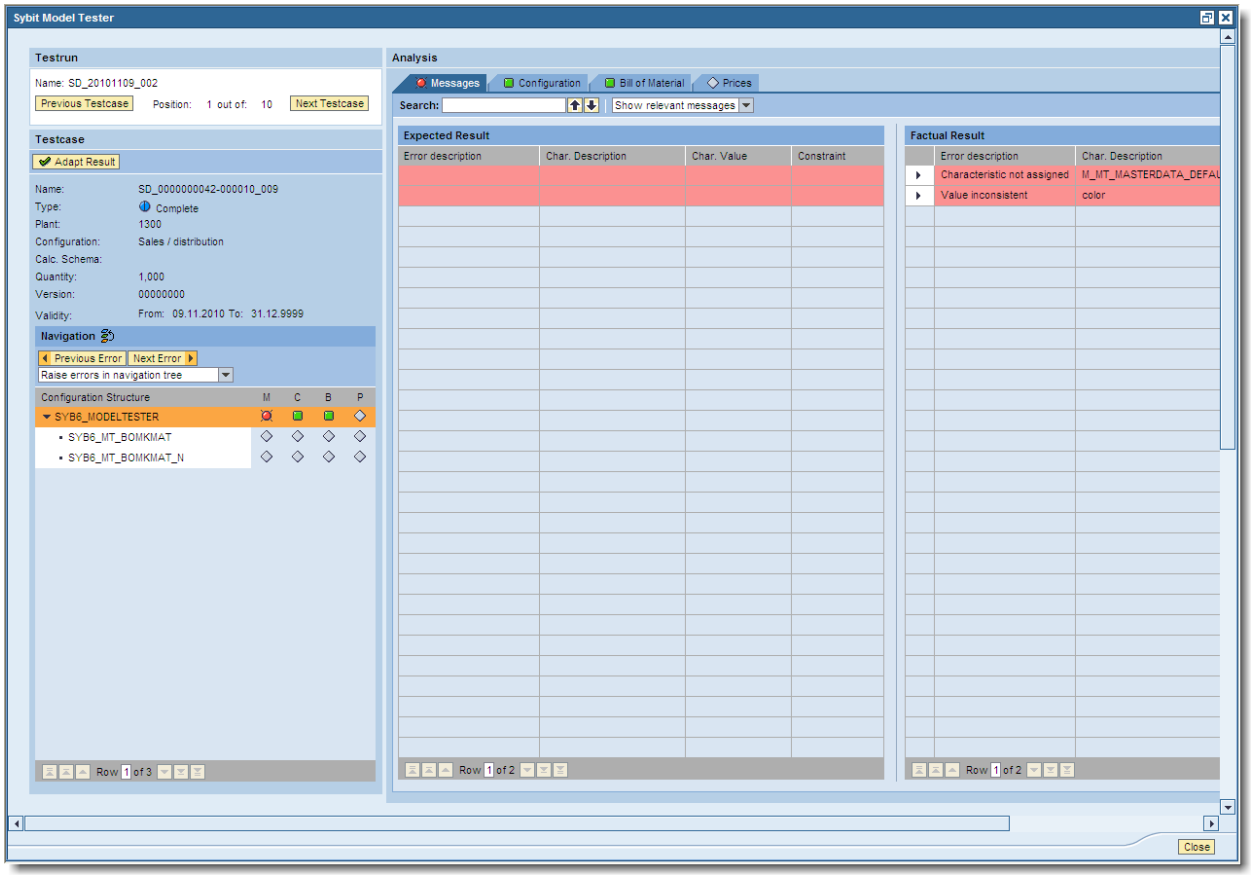


Fig. 4-32: Evaluation tab, analysis popup: Navigation area and messages tab.

4.11.9 Analysis Window: Configuration Tab

Fig. 4-33 shows the configuration tab. Here you see only those characteristics that are assigned with an actual value by the user or by the system – whereas the test case tab in complete view mode also shows characteristics with no values. In the table on the left there is an additional column titled **State**. It tells you if the value has already been part of the input configuration of the test case. A tool tip on the symbol gives you detailed information about the value in the input configuration. You may choose between the following options using the selection area beside the search field:

- **Show conflicts**: Only the lines that contain conflicts (red background) are displayed.
- **Show relevant cstics** (default value): All characteristics that are marked relevant in the profile are shown. Lines with conflicts have red background. If the test run has been executed without any profile all characteristics are shown.
- **Show all cstics**: All characteristics are shown including those that have been marked irrelevant in the profile. Irrelevant characteristics have dark blue background.

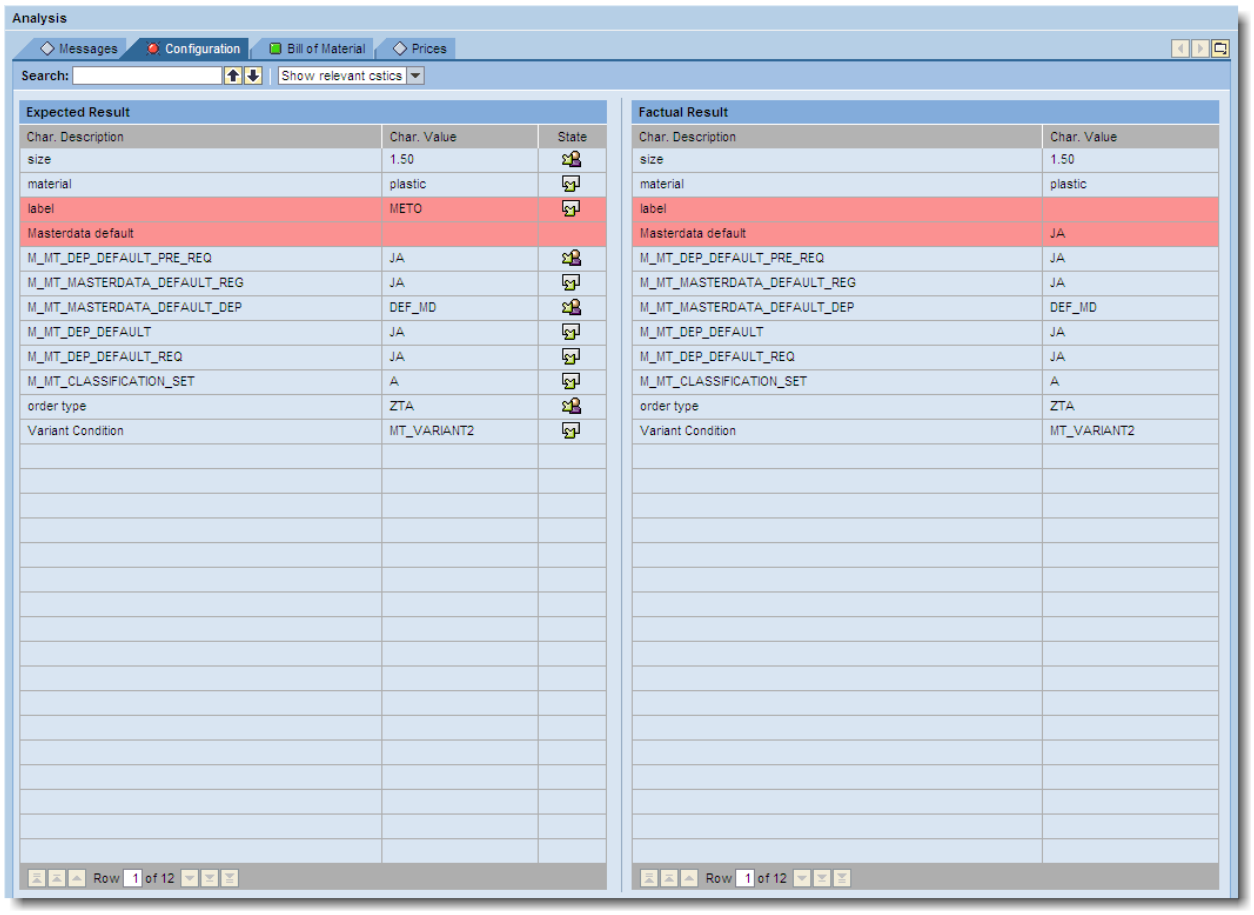


Fig. 4-33: Evaluation tab, analysis popup: Configuration tab.

4.11.10 Analysis Window: Bill of Materials Tab

The BOM tab (Fig. 4-34) corresponds in its structure and functionality to the tabs for messages and configuration, see above. You may choose between the following options using the selection area besides the search field:

- **Show conflicts**: Only the BOM lines that contain conflicts (red background) are displayed.
- **Show relevant data**: All BOM columns that are marked relevant in the profile are displayed. Conflicting lines have red background. If the test run has been executed without any profile all columns are shown.
- **Show all data**: All BOM columns are shown including those that have been marked irrelevant in the profile. Irrelevant columns have dark blue background.

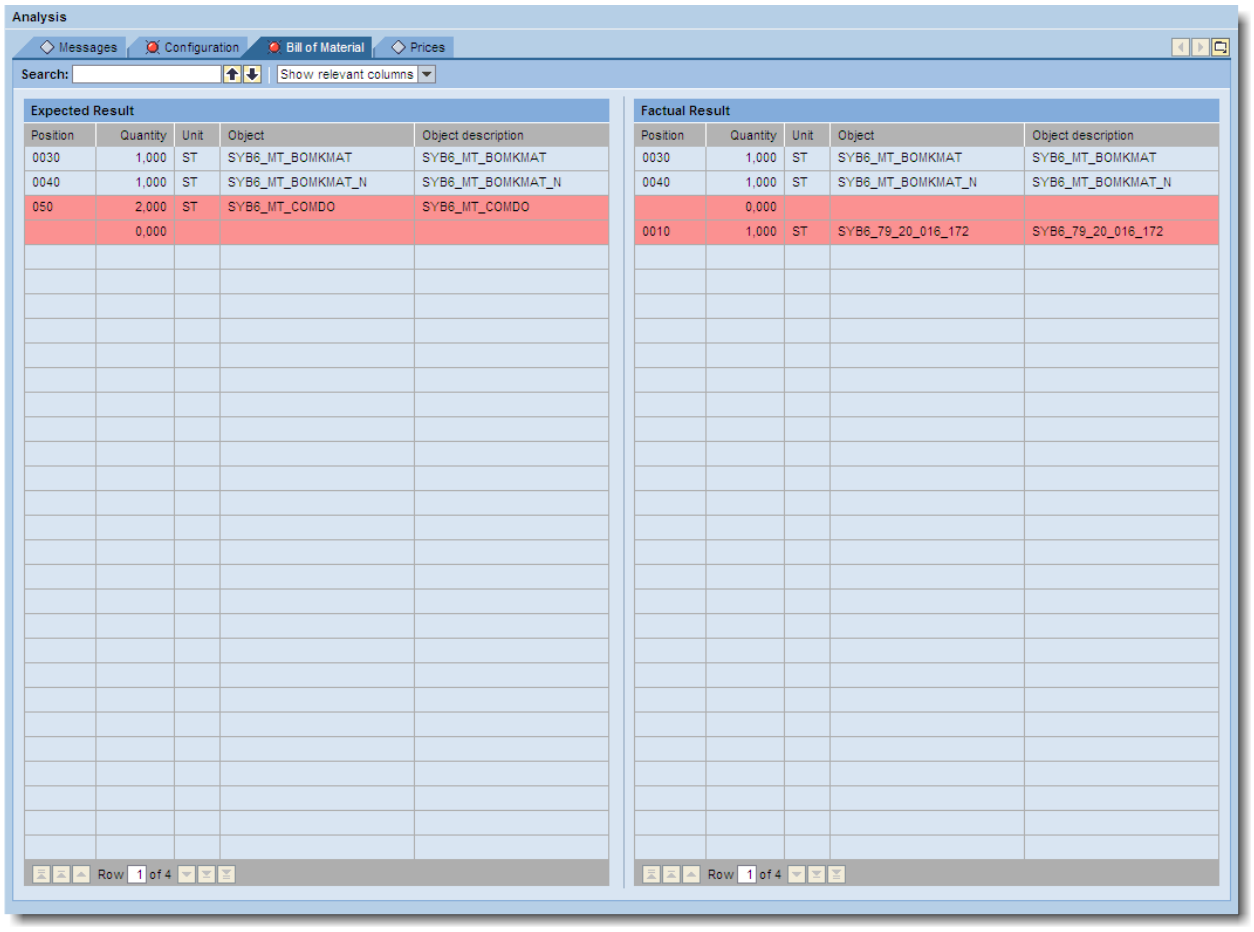


Fig. 4-34: Evaluation tab, analysis popup: BOM tab.

4.11.11 Analysis Window: Prices Tab

The prices tab (Fig. 4-35) complies with the above described tabs concerning structure and functionality. You may choose among the following display modes using the drop down box besides the search field:

- **Show conflicts**: Only the condition records that contain conflicts (red background) are displayed.
- **Show relevant condition records**: All columns that are marked relevant in the profile are displayed. Conflicting lines have red background. If the test run has been executed in an LO-VC and without any profile all columns are shown. In the IPC, the factor column is hidden, see Section 2.9.
- **Show all condition records**: All columns are shown including those that have been marked irrelevant in the profile. Irrelevant columns have dark blue background

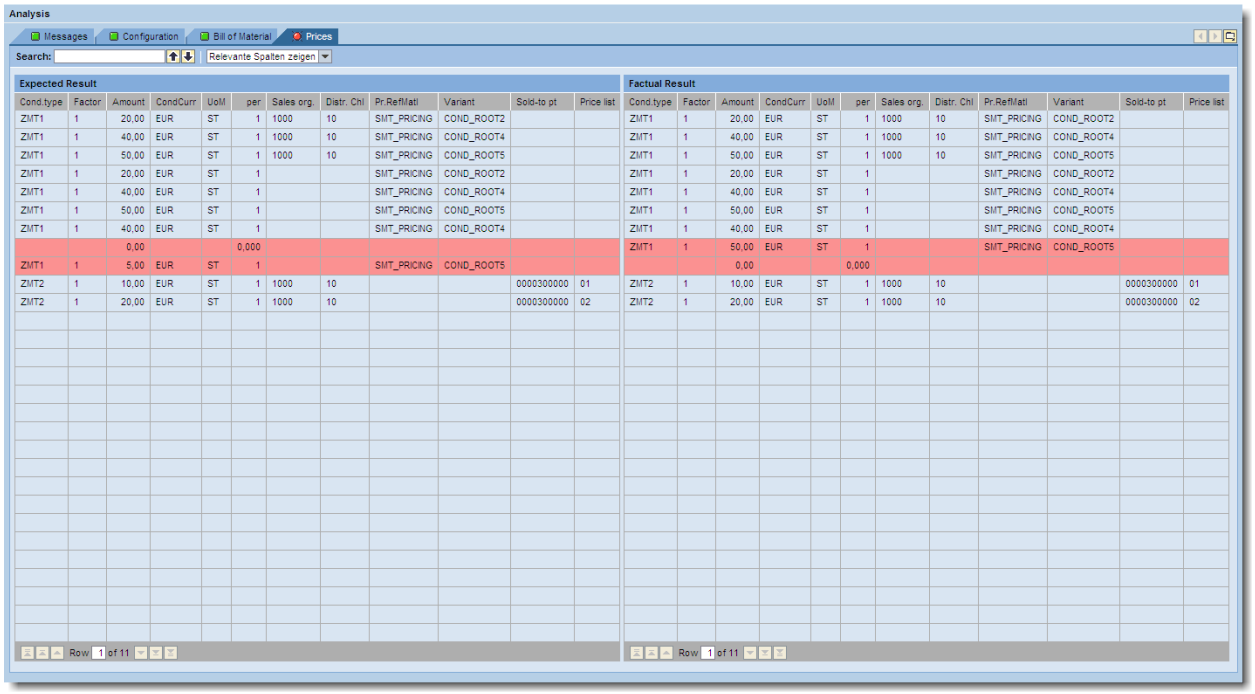


Fig. 4-35: Evaluation tab, analysis popup: Prices tab

4.12 Mass Maintenance

The mass maintenance tab (see Fig. 4-37) contains a tool that allows you to adapt a large number of test cases at once. Since the previous implementation was technologically limited, the tool has been redesigned and enhanced substantially in version 2.1.

The tool is designed primarily to organize configuration model evolvement in the according test set. The tab itself contains a table of test cases you want to edit. The editing itself is done via a set of wizards in a popup window.

4.12.1 Mass Maintenance of Multi Level Test Cases

For the maintenance of multi-level test cases please note the following: Each test case for a multi-level KMat can be divided into a root-item for the main KMat and into a number of sub-items, one for each sub-KMat, see Section 2.5.

Example: Consider a car as a multi-level KMat that contains a seat KMat. Then the car is the root KMat, the seat is a sub-KMat.

If you want to mass maintain a set of multi-level test cases you have to set the KMat that actually changed into the KMat field in the header.

Example: If the configuration model of the seat gets a new required characteristic, you have to enter the seat-KMat into the global KMat field. Then, you can change all test cases that contain the seat as a main KMat or as a sub-KMat in one single action.

To clarify on which test cases you are actually working, test cases for sub-KMats (i.e.: sub-items) inherit the test case name of their main KMat (i.e.: root-items). In the description field of the sub-items you can find the path where they are located in the root-item.

Consider the example in Fig. 4-36. Here, we were searching for tests of the sub-KMat SMT_ABS_TR_MODUL and we did the following:

- In the name field we input the name of a root-item that tests a main KMat.
- In the KMat field we deleted the KMat set by default such we find test cases for all KMats
- In the Level field we changed the default value of root-item to any such that we not only get root-items, but also sub-items that belong to the searched test case.

To make the display more concise we hide all irrelevant columns in the figure. Read more about customizing table columns in Section 4.1.2. From the description of the test cases – that is always maintained automatically by the system – you may get at which position each sub-item is placed inside the multi level hierarchy.

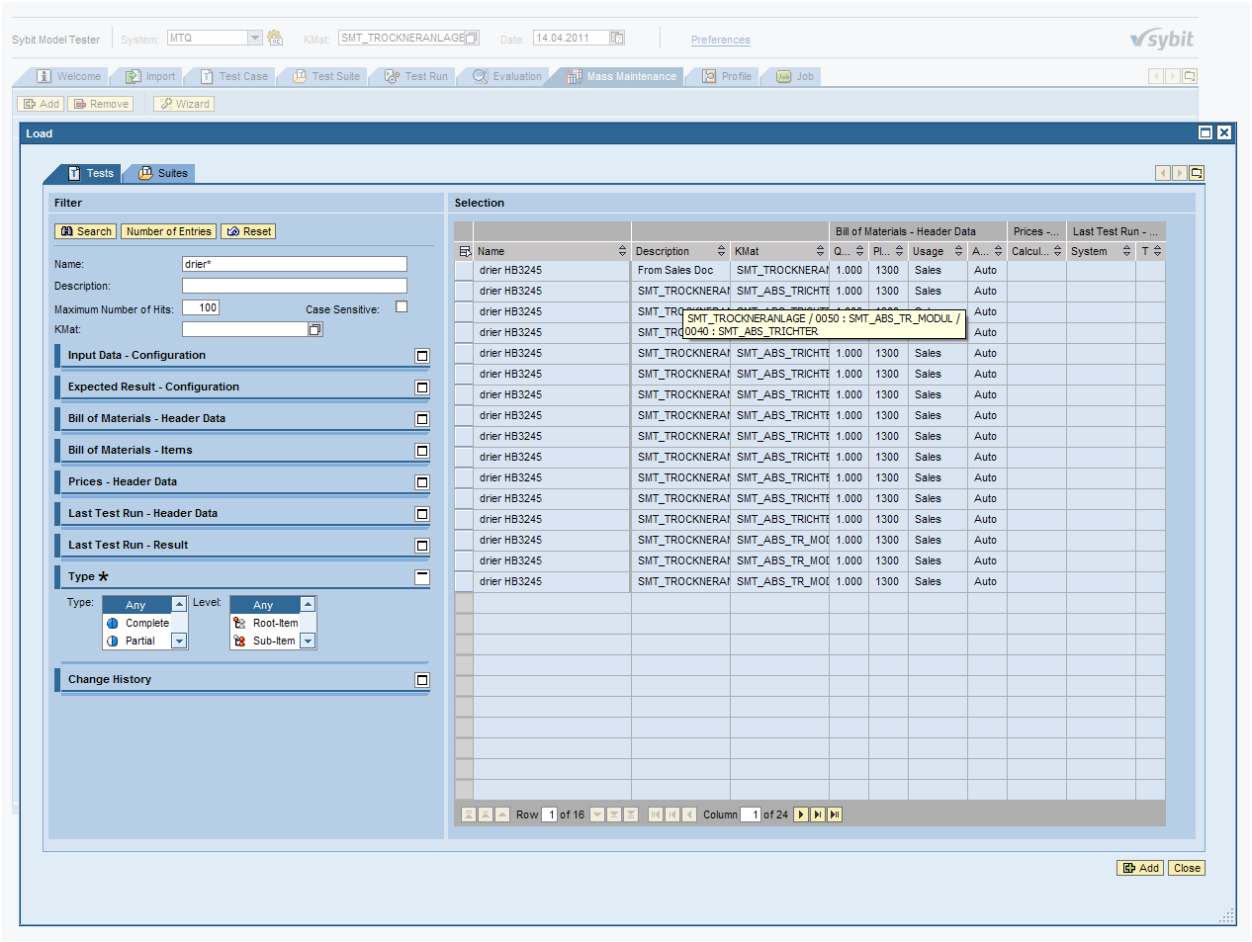


Fig. 4-36: Loading of root-items and sub-items of a test case.

4.12.2 Main View

The main view of the mass maintenance tab is displayed in Figure

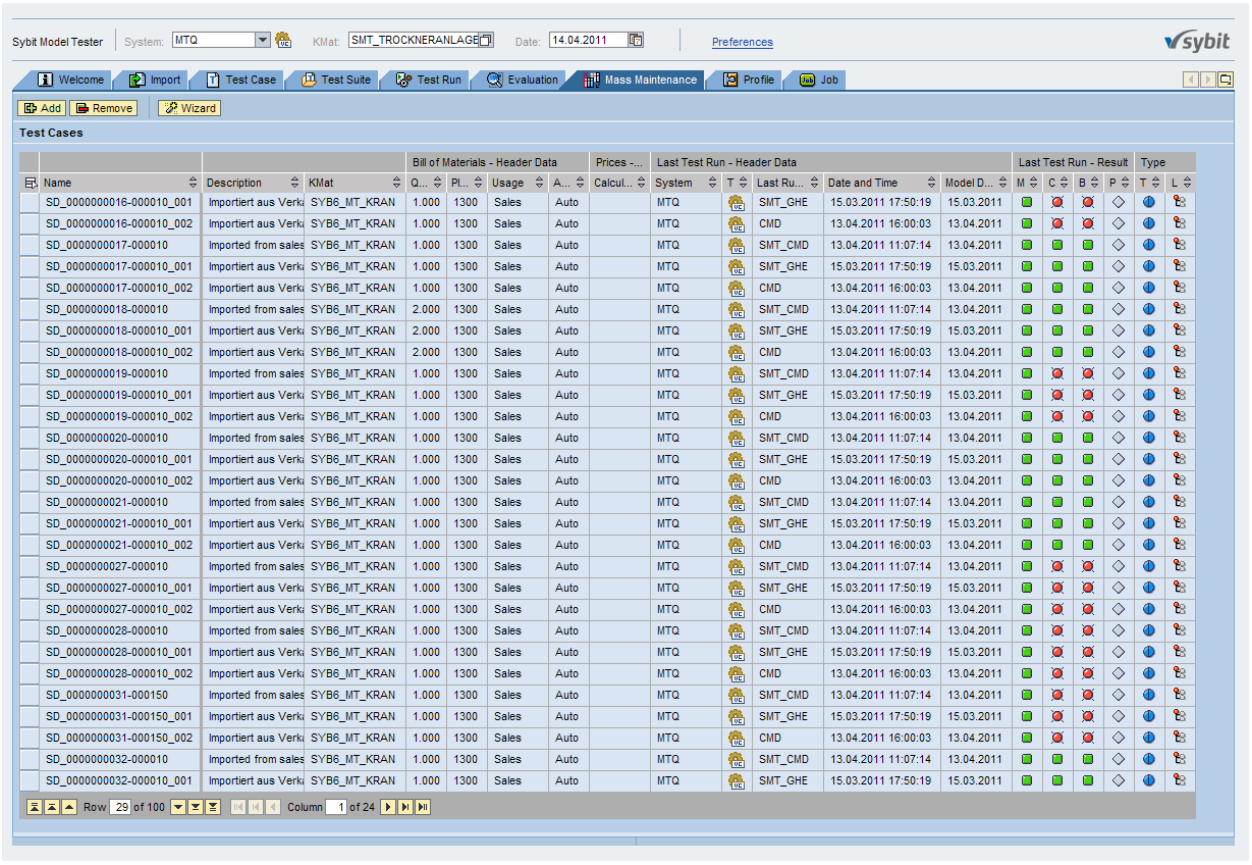


Fig. 4-37: Mass maintenance of test cases.

The main table is you work basket that contains the test cases you want to edit.

In the tool, the following controls are available:

- Pushbuttons for **Add** and **Remove:**
  In a first step you have to define which test cases you want to edit. With the Add / Remove buttons you may add / remove test cases to / from your work basket. To add test cases from multiple searches to the work basket, you may press the Add button repeatedly. Each click on Add opens a popup window that contains the common search component as described in Section 4.5.
  Please notice that you do not have to close the window after each search request. If you click on Add inside the popup window the selected test cases are added to your work basket in the background. Due to technical limitations this becomes visible only after closing the popup window.

- Pushbutton **Wizard**:
  After loading the test cases you want to edit into your working basket, click on the Wizard button to start editing. In a first step the Sybit Model Tester sets a write lock on all test cases in the working basket – comparable to switching on the Edit Mode in the test case tab. This ensures for all displayed test cases that nobody edits them in parallel and that the changes you are about to define can be applied eventually. If the setting the write lock is successful, a popup window opens where you may specify the type of wiz-

ard you want to execute. If test cases are locked by other users, you will get a warning message. Locked test cases will be selected in the table.

If some specific columns in the main view are irrelevant for your requirements, you may just hide them as described in Section 4.5 in the part on hiding table columns.

To simplify the navigation, in the table the test case names have underlying quick links. A click on a test case name opens a popup area that you may use to directly navigate to the test case or the evaluation tab, see

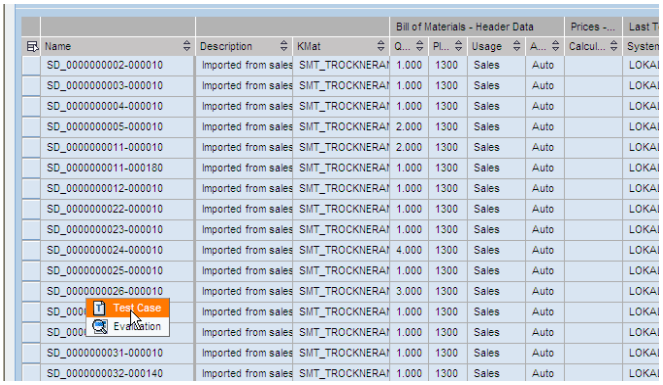


Fig. 4-38: Mass maintenance: Navigation support.

4.12.3 Wizard Selection Window

The window for the selection of a wizard is displayed in Fig. 4-39. The ordering of the individual wizards complies with the structure of a test case. Accordingly, the options **Copy**, **Delete**, **Change Header Data,** and **Repair** operate on the test case as such. If you want to edit a specific part of the test case you may choose between **Input Data** and **Expected Result.** Where in the input data section you may work on the **Configuration** only, the expected result contains areas for **Configuration, Bill of Materials** and **Prices.** If you click on the lowest level of the hierarchy – which is, in the input data e.g. **Set** and **Delete** you will get the according assistant to edit the data.

The Sybit Model Tester contains a large number of different wizards in the mass maintenance tab. Each wizard is targeted towards one specific aspect of test cases. If you want to edit test cases in more than one aspect you have to call different wizards consecutively. The test case selection in your working basket sustains – as long as you do not change it explicitly.

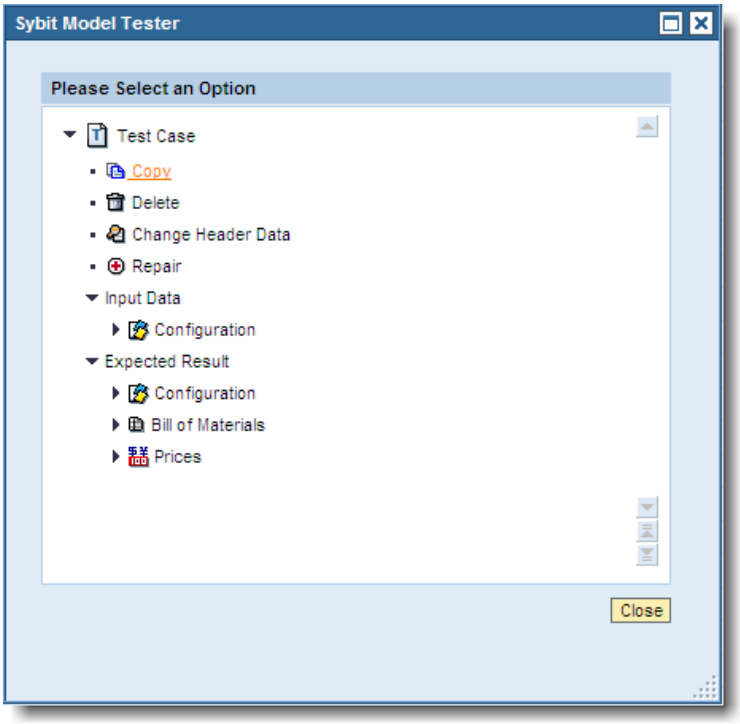


Fig. 4-39: Mass maintenance: Wizard selection window.

4.12.4 Wizard: Common Graphical Structure

In this section we do not go into detail on each single wizard. In contrast we explain the common principles that are shared by all wizards. As an example, Fig. 4-40 shows one step of a wizard.

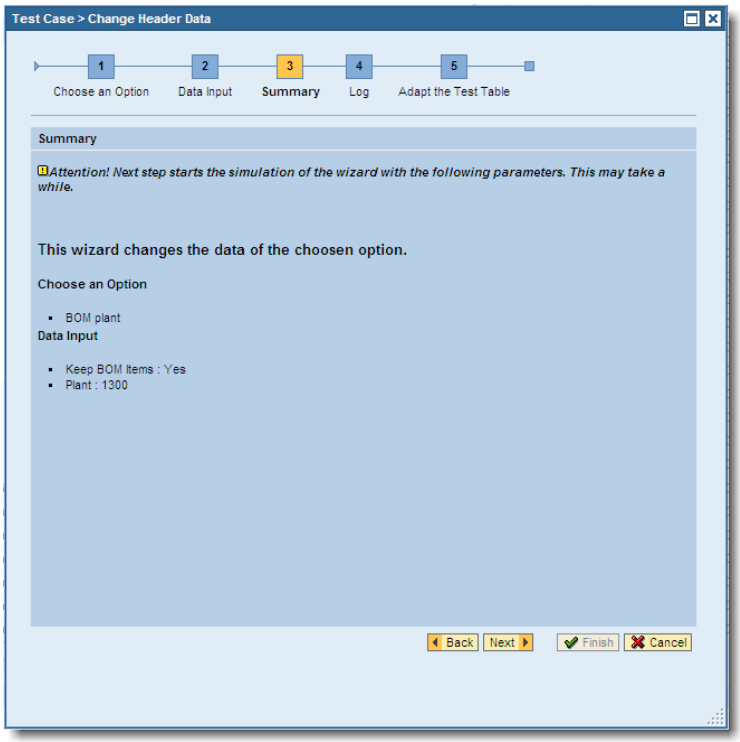


Fig. 4-40: Mass maintenance: Wizard.

All wizards are divided into the following three areas

- **Header area with step sequence**: The header area contains the so-called step sequence where you get an overview of the steps in the wizard. The step currently active step is highlighted and written in bold font.
- **Main area**: Here you input your data and check their correctness.
- **Footer area with pushbuttons**:
  - **Back**: Advances to the next step of the wizard.
  - **Next**: Returns to the previous step of the wizard.
  - **Finish**: Writes the changes irrevocably into the test case database. This pushbutton is only active in the steps **Log** and **Adapt Test Table**.
  - **Cancel**: Cancels the wizard. Test cases are not changed.

4.12.5 Wizards: Common Steps

All wizards follow the same step sequence schema:

- **Data input**: At the beginning, each wizard contains a number of steps (at least one) where you input data.
- **Summary**: The third last step in each wizard is the summary screen. Here the user input data is summarized on one single screen. Please use this screen to check the validity of you input data.
  If you click on **Next** in the Summary step the Sybit Model Tester tries to apply your changes to the test cases in your working basket. However, the changes are only *simulated* and *not* written to the data base.
- **Log**: The second last step shows the log of the simulation. The messages are marked with colored icons. The following colors occur:
  - **Green**: The specified change can be applied to the test case.
  - **Yellow**: The specified change can not be applied to the test case, since the test case already conforms to the desired result of that change.
    Example: For all test cases you want to assign the value XXL to the characteristic M_SIZE. Then, test cases that already contain M_SIZE = XXL get yellow icons.
  - **Red**: The specified changes could not be applied to the test case.

  If it is necessary to execute more than one action for a wizard, you will get one message for each action. Hence, in these cases you will get multiple messages for each test case with possibly different colors in the log.
  - A click on **Finish** writes the results of the simulation irrevocably into the test case database. Subsequently the wizard is closed automatically.
  - If you click on **Next** in Log step the results are still not written to the test case database but the wizard takes you to its last step.
- **Adapt the Test Table**: In this step you may adjust the set of test cases in your working basket in the main view of the tab – based on the results of the log step.
  A click on **Finish** writes the results of the simulation irrevocably into the test case database. Subsequently the wizard is closed automatically.
  The pushbutton **Next** is disabled in this step.

4.12.6 Wizard: Filter – Logic

Many wizards – as e.g. the one to change items on the bill of materials – share the same logic concerning the definition of filters.

- **Define a filter**: In this step you specify the elements of the test case to which you want to apply your changes, see Fig. 4-41. Empty entries in filter fields for string values or zero value entries in number fields are treated as wildcards, i.e. they match any value. In cases where the wizard explicitly allows filtering for zero values (as e.g. the wizard to change prices, see Fig. 4-42) you may activate filter lines individually.

  Example 1: You want to exchange the screw SCREW_123K45 at position 0320 on the bill of materials of all test cases with the screw SCREW_FE9999 of a different supplier. Your filter looks as follows:
  Component. SCREW_123K45
  Position: 0320
  Quantity: 1
  Unit: PC

  Example 2: Unlike example 1 you not only want to change the screw at this position but all screws of this type. Then your filter looks as follows:
  Component. SCREW_123K45
  Position:
  Quantity: 0.000 (unchanged default value)
  Unit:

  With this filter the wizard will adapt all positions on the bill of materials that contain the specified screw regardless of their position, their quantity, or their unit.

- **Specify new values**: The New Values screen lets you specify the values you want to set in all places defined by your filter see Fig. 4-43. Empty values in input fields for strings or zero-valued entries in input fields for numbers mean that the respective value is not adjusted.

  Example 1: As in examples 1 and 2 above, we want to exchange the screw. To realize this you need to specify the following values
  Component: SCREW_FE9999
  Position:
  Quantity: 0.000 (unchanged default value)
  Unit:

  Example 2: In the example 1 above you want to double up each screw with an extra screw. Then you need to specify:
  Component: SCREW_FE9999
  Position:
  Quantity: 2.000
  Unit:

  For example 2 above this does not work necessarily since you would set the quantity of the screw to 2.000 regardless of where it appears in the bill of materials.

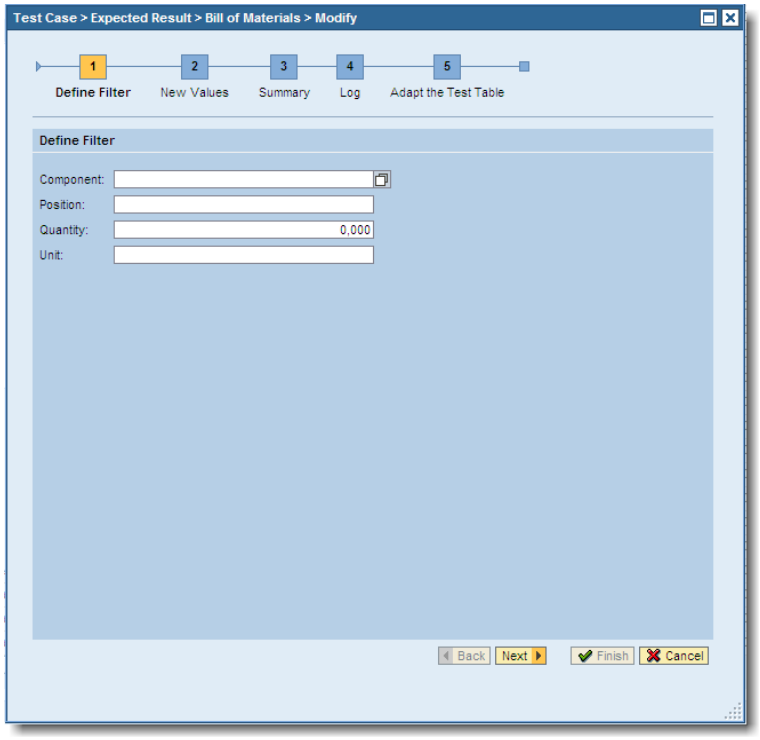


Fig. 4-41: Mass maintenance: Defining a filter for changing the bill of materials.

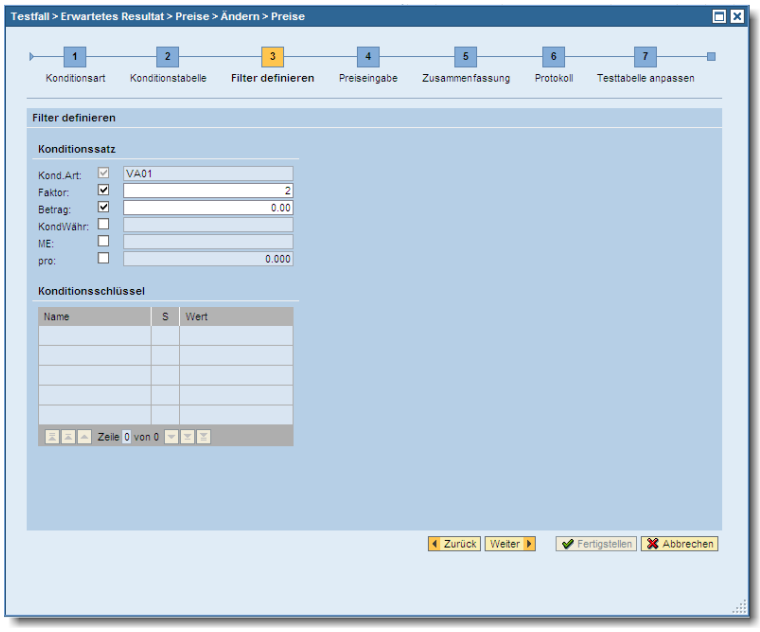


Fig. 4-42: Mass maintenance: Defining a filter to change prices.

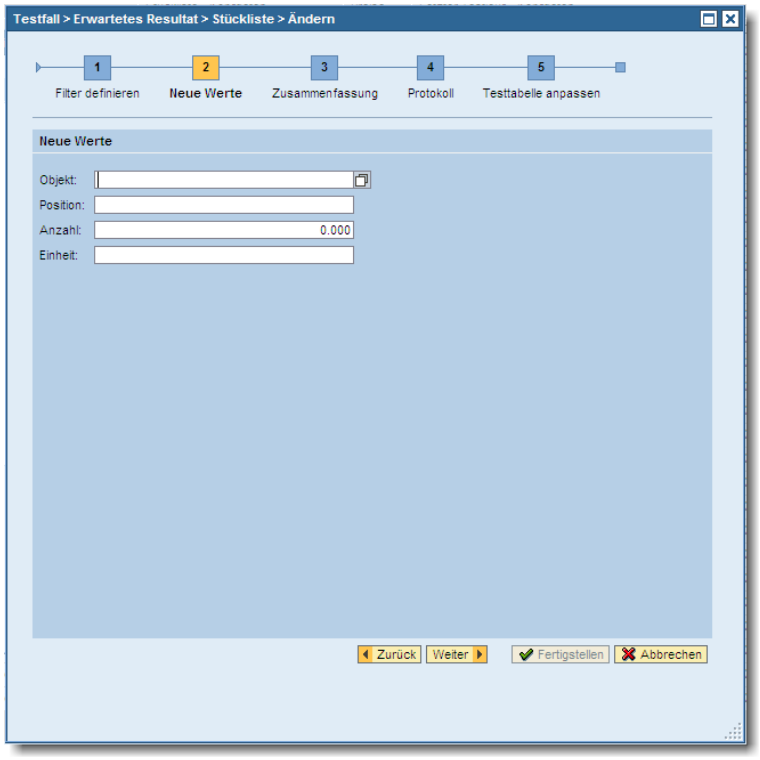


Fig. 4-43: Mass maintenance: Setting a new value on the bill of materials.

4.13 Profile

The profile tab is used to create and maintain profiles. The Sybit Model Tester uses profiles to fine-tune the behavior of test runs and their evaluation, respectively. Currently, the main task of profiles is to mask – i.e. to hide – foreseeable errors in the evaluation.

Example: Depending on the usage scenario, certain system set characteristics might be irrelevant for the evaluation of the test run. E.g. there are characteristics that have only a model internal meaning or there are characteristics that have been made dispensable by model evolution but can not be deleted for technical reasons. If you are not interested in their values, it makes no sense to look at them at all. To this end, these characteristics can be marked to be irrelevant in the profile. If in a test run the expected characteristic value does not match the factual characteristic value, the test case is successful if and only if this characteristic has been marked irrelevant. Another scenario might be that you are not interested in testing the BOM explosion at all, or that for your evaluation the matching of BOM position numbers is not relevant.

Similar to the expected and the factual result, the profile is structured in areas: The messages area, the configuration area, the BOM, and the pricing area. Fig. 4-44 shows the profile tab with its messages area.

In contrast to the tabs for test case, test suite, and test run, the profile tab is independent of the KMat in the header area. Each profile can contain any number of KMats. This implies that you only need one profile for multi-level test cases. Another application for many KMats in a profile is that you may specify one single central company specific profile that defines the relevance of all characteristics in all KMats under test.

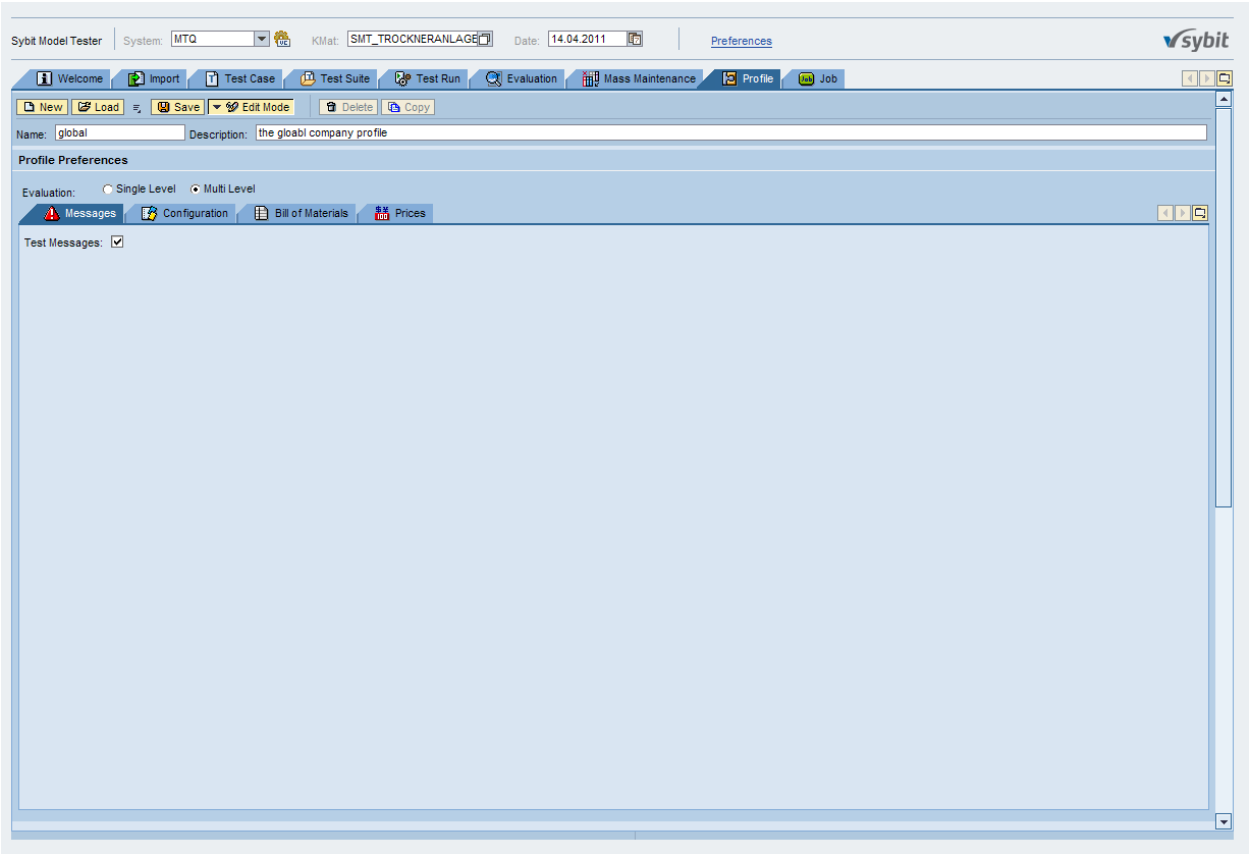


Fig. 4-44: Profile tab: Messages tab.

The following controls are available in the tool:

- **New** / **Load** / **Save** / **Edit Mode** / **Delete** / **Copy**: Similar as in the test case tab, see Section 4.8.
- **Name** / **Description**: Name and description of the Profile. Again similar as in the test case tab, see Section 4.8.
- **Evaluation single level / multi level:** Via this radio button you may specify if test cases that are run with this profile will be evaluated as a single level KMat or as a multi level KMat. If your configuration model is single level, this setting has no effect on your test run. If you set this switch to single level in while testing a multi level configuration model only the root-item / root-KMat will be considered relevant for the evaluation. Sub-items will be treated irrelevant. If the sub-item of a test case would fail in a test run, this will not affect the test run if you run it with a profile with single level evaluation set.

4.13.1 Irrelevant areas – Basics

In a profile, you may specify for each test case area individually if the area will be marked as relevant or irrelevant.

Example: The option “Test Configuration” in the configuration tab specifies if the comparison between expected configuration and factual configuration is executed as a whole. However, this option does *not* influence the messages area. I.e. if the “Test Messages” flag is set in the messages tab, all messages are displayed and evaluated. This includes messages that are triggered by problems in the configuration area such as “Characteristic not assigned”, “Characteristic inconsistent”, etc. The

option “Test Prices” in the prices tab behaves similarly for messages from the pricing area such as “Condition not found”, etc.

If you want to ignore messages for a test run, you have to deactivate the “Test Messages” flag in the messages tab.

An exception of this behavior is if you set specific characteristics individually to be irrelevant in the configuration tab. This function does not only affect the configuration but also the messages area. Hence, all messages that belong to a characteristic marked as irrelevant will also be treated as irrelevant. The reason behind this is the following: If a user explicitly states to be not interested in a certain characteristic then in general he will not be interested in any message concerning this characteristic.

4.13.2 Messages Tab

In the current version, the messages tab lets you switch on or off messages, see Fig. 4-44.

4.13.3 Configuration Tab

In the configuration tab you may specify how the Model Tester deals with conflicts in the configuration area, see Fig. 4-45. Here, the main purpose is to hide and disregard predictable conflicts for certain characteristics. For any number of KMats you may specify a list of irrelevant characteristics. If a conflict occurs in an irrelevant characteristic, it will not lead to a failure of the test case.

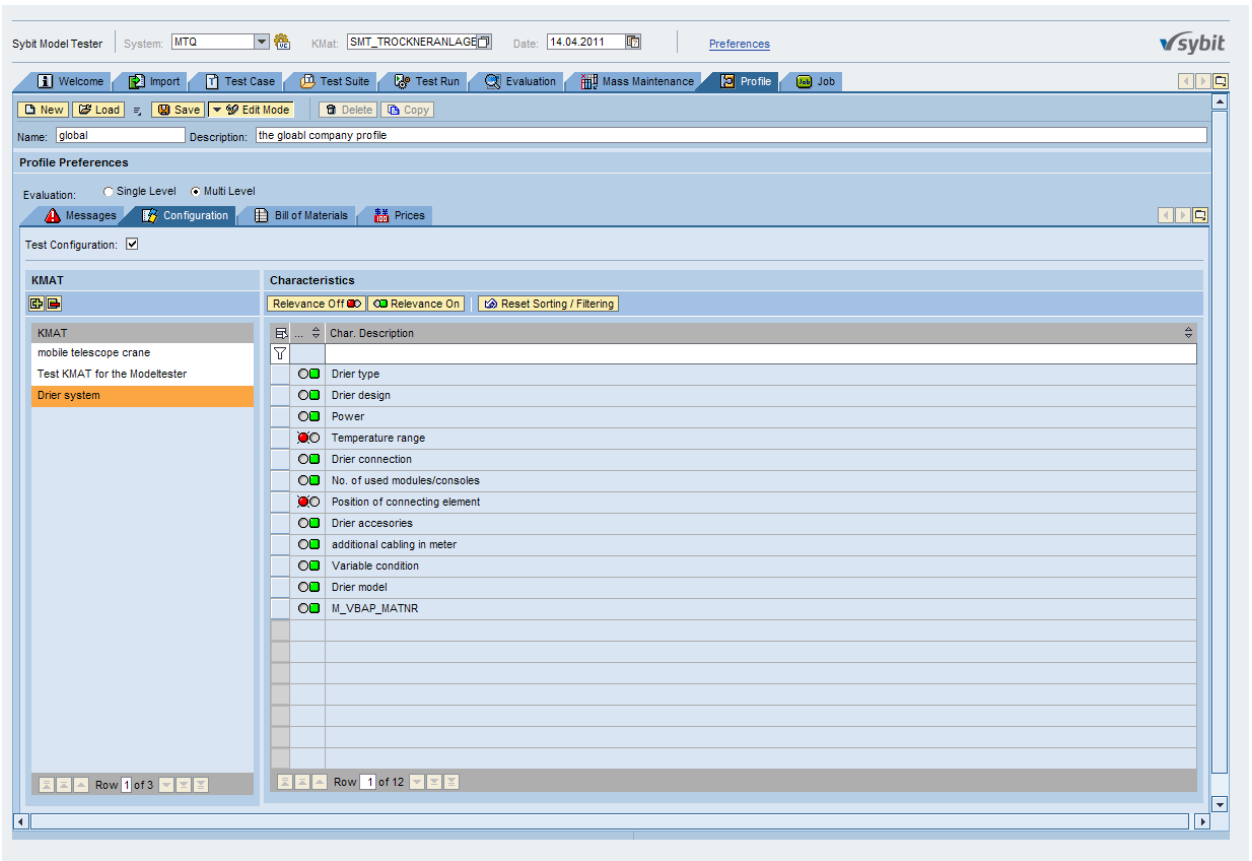


Fig. 4-45: Profile tab: Configuration tab.

The following controls are available:

- **Test configuration**: Use this check box to specify if the configuration area is relevant for you or not *as a whole*. If you uncheck this check box you will never get an error in the configuration area.

KMat table: Shows the list of all KMats for which you may specify irrelevant characteristics in this profile.

- **Add KMat** - Button: Opens a search help window where you may choose a KMat to be inserted into the table.
- **Delete KMat** - Button: Deletes the selected KMat from the profile.

Characteristics-Table: Use this table to define the relevance of characteristics contained by the KMat that is selected in the KMat table.

- **Relevance off**: All selected characteristics are marked irrelevant.
- **Relevance on:** All selected characteristics are marked relevant.
- **Reset sorting:** If you sorted the list of characteristics according to their status, this button resets the sorting. I.e. characteristics are displayed according to their class order.
- **Filter line**: In this line you may specify a filter expression to restrict the displayed characteristics to a suitable subset.

4.13.4 Bill of Materials Tab

In the BOM tab you may specify how the Model Tester deals with conflicts in the BOM area, see Fig. 4-46. Here, the main purpose is to hide and disregard predictable conflicts for certain BOM columns. If a conflict occurs in an irrelevant column, it will not lead to a failure of the test case.

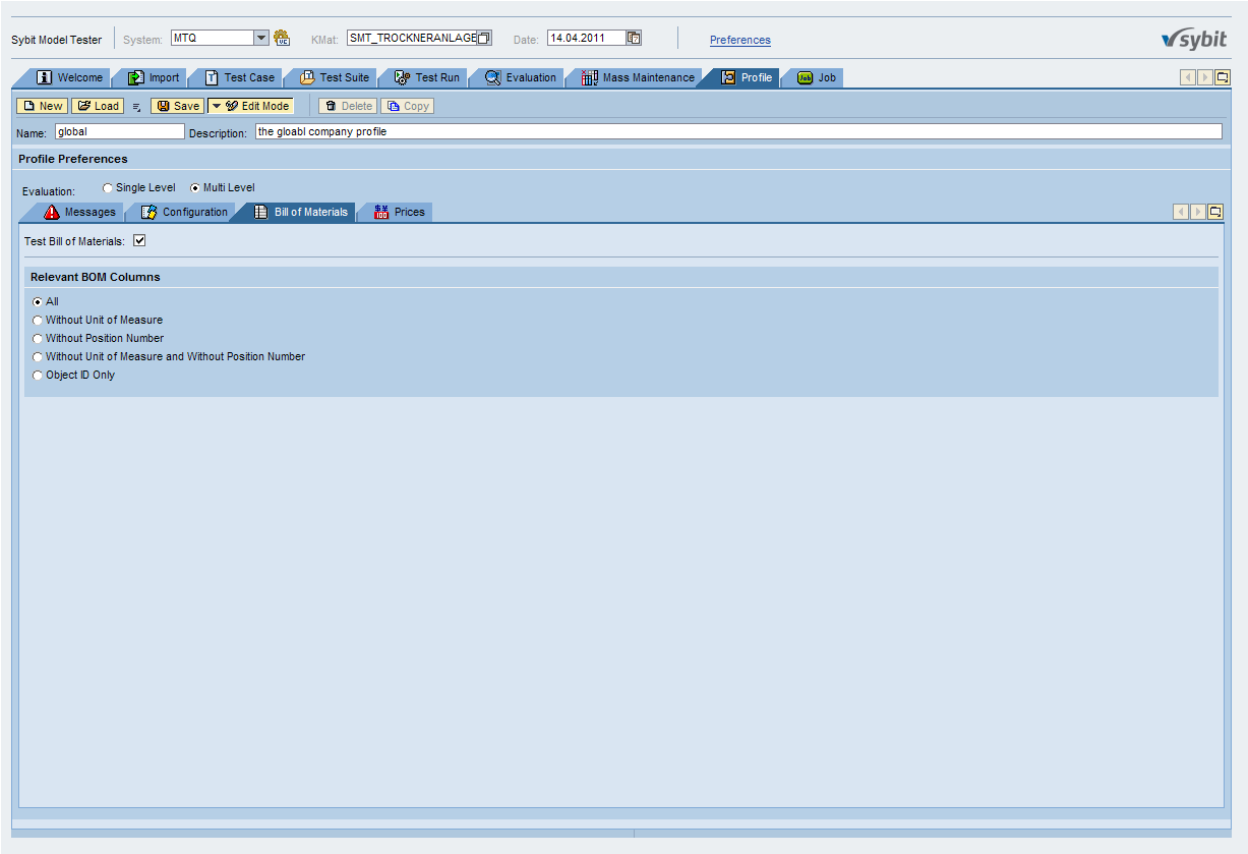


Fig. 4-46: Profile tab: BOM tab.

The following controls are available:

- **Test expected bill of materials**: Use this check box to specify if the BOM area is relevant for you or not *as a whole*. If you uncheck this check box the test run will disregard BOM explosion. Hence the BOM is not tested at all.
- **Relevant BOM columns**: Here you can define which columns of the BOM are relevant. In the Model Tester, the bill of materials contains the following columns: Position, quantity, unit, object ID (i.e. material ID), and language dependent description of the material. Since the Model Tester always relies on language independent IDs, the language dependent description is *never* relevant. The following options are available:
  - **All**: All columns are relevant for evaluation.
  - **W/o unit**: The unit is irrelevant. The position, the quantity, and the object ID are relevant.

- **W/o position number**: The position number is irrelevant. The unit, the quantity, and the object ID are relevant.
- **W/o unit, position number**: The unit and the position number are irrelevant. The quantity and the object ID are relevant.
- **Object ID only**: The unit, the position number, and the quantity are irrelevant. Only the object ID is relevant for evaluation.

4.13.5 Prices Tab

In this area you specify how to deal with errors in the prices area of test cases, see Fig. 4-47. You may define which columns of the prices table should be relevant for the success of test run. An error in a column that is marked as irrelevant will not lead to a failure of the test case.

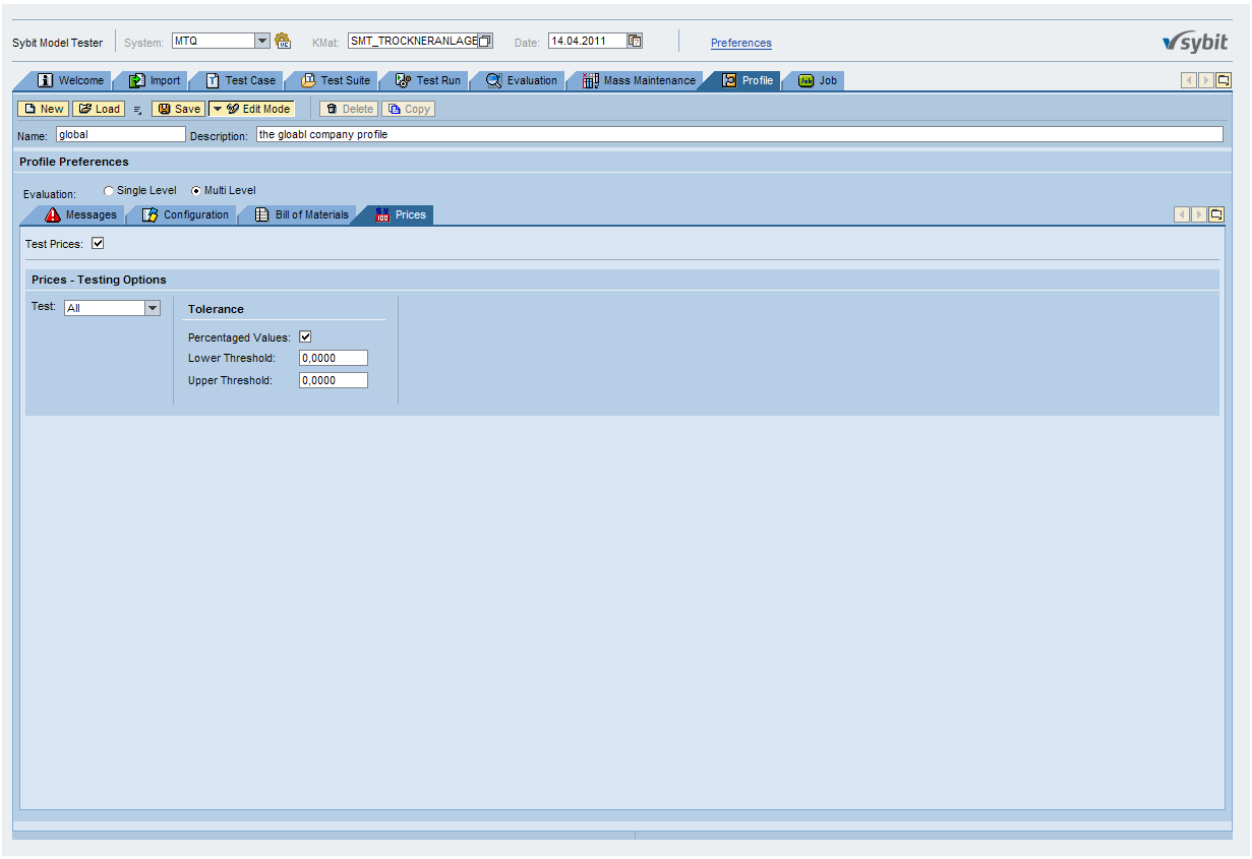


Fig. 4-47: Profile tab: Prices tab.

The following controls are available:

- **Test Prices:** This check box determines if the prices area is tested or not as a whole. If you uncheck this check box no prices will be tested.
- **Test Amount:** In this software version, you may choose among two options:
  - **All:** The analysis of a test run considers all columns to be relevant.
  - **W/o amount, currency:** In many scenarios, prices change so quickly that the effort of maintaining prices in test cases would not pay. If you choose this option, the system checks the existence of condition records only.

- **Tolerance:** If you choose to test **All** in the drop down box, you may specify a tolerance area for testing amounts. Using the input fields for **Low tolerance** and **High tolerance**, you may specify how much the amount in the factual result may differ from the amount in the expected result without causing the test to fail. If you choose **percentage values,** the input values define a tolerance area as percentage of the expected result. Otherwise, the values are treated as absolute amounts.

4.14 Job

The job tab enables you to schedule test runs periodically. We want to illustrate possible usage scenarios of the job functionality by means of two examples.

4.14.1 Example Scenario 1: Changes in the Productive System

In this scenario – that is actually pretty frequent – new configurable materials (KMats) are created in the productive system directly. After releasing them changes to the configuration model are done likewise in the productive system. Often, changes of the configuration model are ordered and implemented on short notice. If you don't have automatic test support you usually avoid the effort to run extensive tests after each model change. Consequently you will eventually get unwanted side effects of your changes that have a bad influence on model quality. In this scenario you model quality decreases steadily.

In this scenario you may drastically improve model quality if you can trigger regression tests fully automatically, e.g. you could execute the same set of test cases every night on your productive system. If we talk about regression tests we refer to a large number of test cases that are assumed to guarantee model quality at any time. In Fig. 4-48 this example is illustrated by Job_1. Job_1 ensures that every night, regression tests are triggered in the currently active configuration model. Bugs in the model can be determined and eliminated quickly.

4.14.2 Example Scenario 2: Engineering Change Management (ECM)

In another widely used scenario model changes are also directly implemented in the productive system but – as regulated by the SAP Engineering Change Management (ECM) – with a logical model date that is far in the future. Changes in the model come into effect only at that date. For this scenario let us assume a fixed release cycle of one month for changes to the configuration model. I.e. only within this cycle model changes are defined, tested, and implemented.

In this scenario the scheduling of jobs can help significantly. As illustrated in Fig. 4-48 by Job_2, you can execute regression test regularly – e.g. every night – but this time with a future logical model date. Hence, the regression test does not test the currently active model, but the model that will be active in the future at the date of the logical model date. If you move – as it is sketched in the figure – the model date into the future compared with the execution date of the test run and choose an offset of exactly one release cycle, you are always testing the next release of you configuration model.

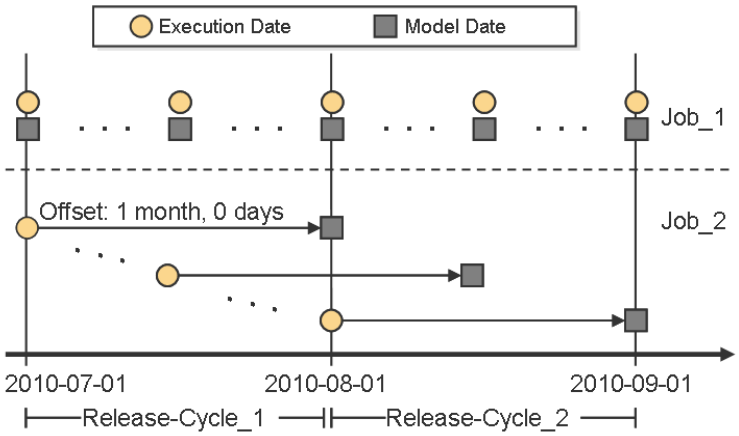


Fig. 4-48: Scheduling of jobs: Two examples.

4.14.3 SAP-Jobs and SMT-Jobs

To realize a job scheduling as described in Sections 4.14.1 and 4.14.2 you have to do two things.

- You have to specify a number of test cases and test suites in the Sybit Model Tester that you define to be you regression tests.
- The scheduling of the job takes place by means of SAP's standard transaction for job scheduling, SM37.

Fig. 4-49 illustrates this relationship. For each job in the Sybit Model Tester – in the following we call this an **SMT-Job**, the system automatically generates a respective job in the SAP system – which we will call **SAP-Job**. SMT-Job and SAP-Job always have identical names.

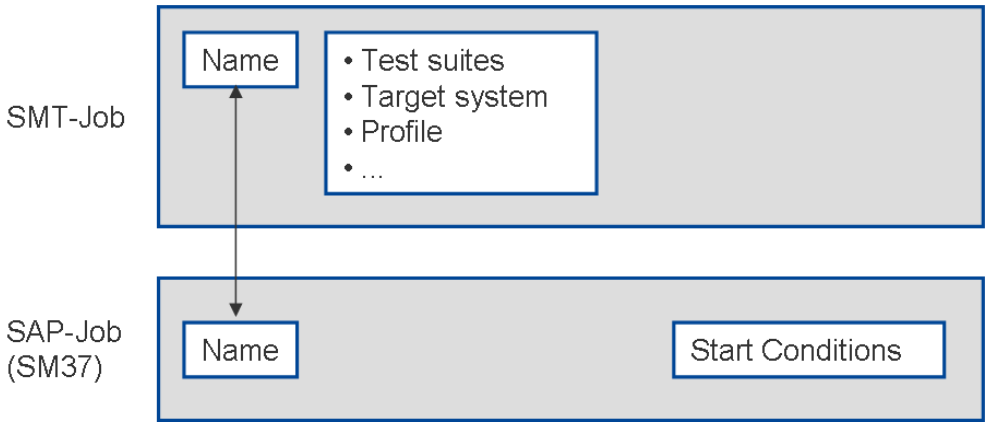


Fig. 4-49: SMT-Job and SAP-Job: Connected via their Names.

The content of the job, i.e. the information which test cases should be executed with which profile and which system is defined in the SMT-Job. The scheduling information is independently stored in the SAP-Job (transaction SM37).

For performance reasons, on execution of a job each contained test suite is executed as a separate test run. The evaluation of those test runs can be done as for all other test runs in the evaluation tab, see Section 4.11.

4.14.4 Editing SMT-Jobs

Fig. 4-50 shows the job tab.

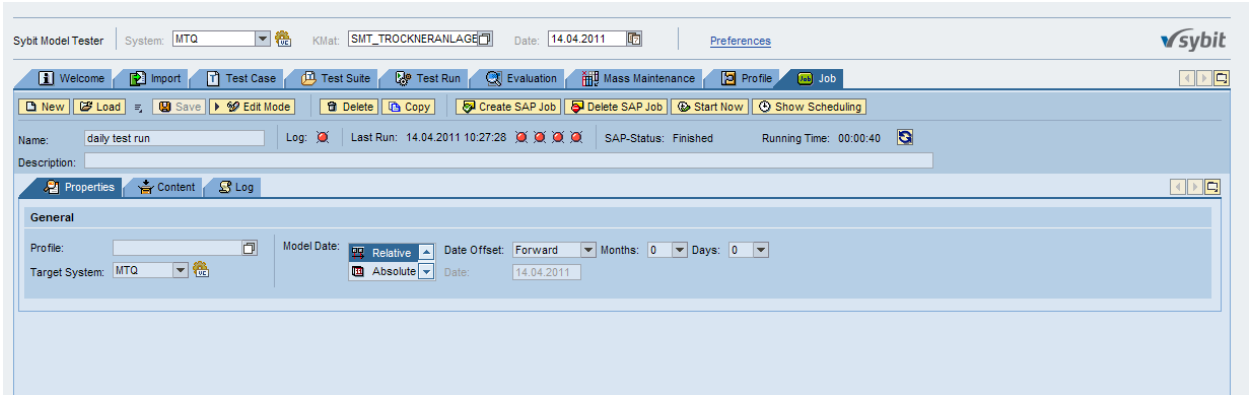


Fig. 4-50: Job tab: Properties tab.

The following controls are available:

- **New / Load / Save / Edit Mode / Delete / Copy**: As in the test case tab (see Section 4.8).
- **Create SAP Job**: This button causes a new SAP-Job to be created in the SAP system. The new SAP-Job has the same name as the SMT-Job. From now on, it can be scheduled via transaction SM37.
- **Delete SAP Job**: This button is the opposite of Create SAP Job. It deletes the appropriate SAP-Job from the SAP system. Data stored in the Sybit Model Tester is not deleted. Only the scheduling information in the SAP system is discarded.
- **Start immediately**: If you press this button, the job you are currently viewing is executed right away. In contrast with a similar button in transaction SM37, this button does not discard the scheduling information of the SAP-Job.
- **Show scheduling**: This button opens a popup dialogue that displays the scheduling information as specified by transaction SM37.
- **Name / Description**: Name and description of the SMT-Job. If an SAP-Job has been created, you can not change the name of the SMT-Job any more. Otherwise you would break the connection between SAP-Job and SMT-Job.
- **Log**: Shows the summarized state of the last run of the job. The icon displayed here equals the icon of the run in the subordinate log tab. First, it contains information about the success of all areas of all test cases in all test runs. Second, it indicates if the job as a whole has been executed successfully.
  Consider the following example: A job contains exactly one test suite. On execution time of the job, one test case of this test suite is being edited and – hence – the test case is locked. It can not be run. All other test cases can be run without conflicts. In this scenario, the icon for the job is red whereas the icons for all areas are green. To learn the reason for the red job icon, the user can check the execution log inside the log tab, see below.
- **Last Run**: Time stamp of the last run and icons for the last result of the job for each area.
- **SAP-Status / Runtime / Refresh button**: Here you can see the status of the SAP-Job. If the job is currently running, the runtime field shows the elapsed time. Otherwise, it shows the total runtime of the last run. The refresh button affects the fields for SAP status and runtime only.

4.14.5 Properties Tab

The job tab contains three sub-tabs. Fig. 4-50 shows the properties tab.

The following controls are available:

- **Profile**: Determines the profile with which the job is executed.
- **System**: Determines the target system on which the job is executed. If your system is an IPC system, please read Section 2.9.
- **Model Date**: If you execute a job periodically then you have to decide if you want the model date to change in relation with the execution date, or if the job always uses the same (absolute) model date.
  - **Relative**: Here, you may specify a relative date which moves alongside the job execution date into the future. To this end, you may specify a date offset and additionally if the **offset** should be added to (**Forward**) or subtracted from (**Backward**) the execution date. Use the input fields for **Months** and **Days** to specify the offset in the range between 0 and 99.

- **Absolute**: If you specify an absolute model date, all executions of this job will use the specified date as model date.

4.14.6 Contents Tab

The contents tab is displayed in Fig. 4-51. In its functionality it is very similar to the test suite tab as described in Section 4.9. Please note the following differences:

- **Test suites:** Different from test suites that may contain test cases only, a job may contain multiple test suites. This takes effect in extended search functionality for the table on the right hand side. To use the extended search, press the according button:
- **KMats:** Test suites may contain test cases for one single KMat only. This restriction is not valid for jobs. To this end, on the left hand side, you may choose any KMat.
- **Navigate to Evaluation:** To analyze test runs, the name of each entry in the table on the right carries a quick link that enables instant navigation into the evaluation tab. Additionally, you may also navigate into the test case or test suite tab, respectively. To make the navigation links visible, you just have to click on the name column of the entry.

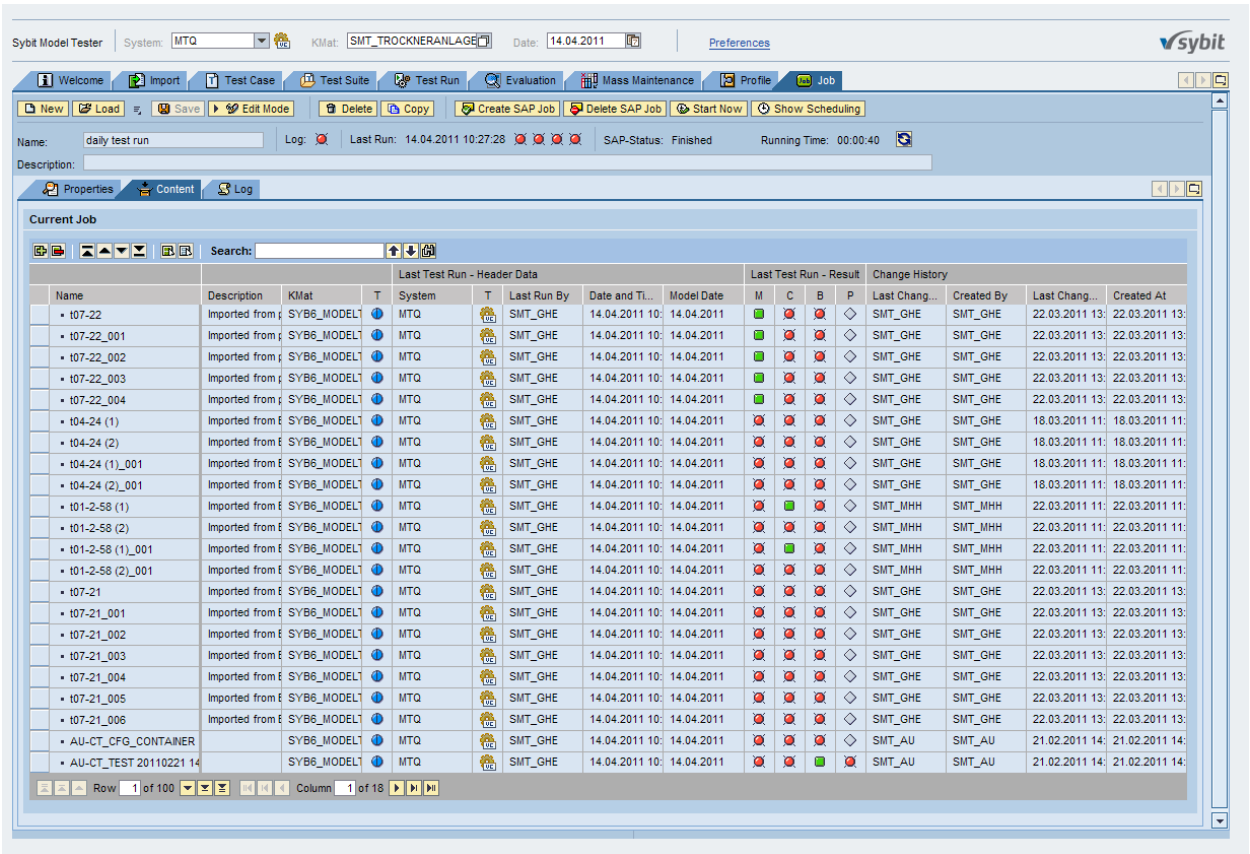


Fig. 4-51: Job tab: Content tab.

4.14.7 Log Tab

The log tab is displayed in Fig. 4-52. It offers a view on information that has been generated automatically during the execution of the job. In the upper **Filter** area you may specify a time span in which you want to search for logs. The **Log** area on the left side contains a **refresh** button that fills the log table below with current information from the system. After pressing refresh, the table con-

tains the logs of all job executions within the specified time span. A click on a time stamp in the table loads the respective log into the table on the right.

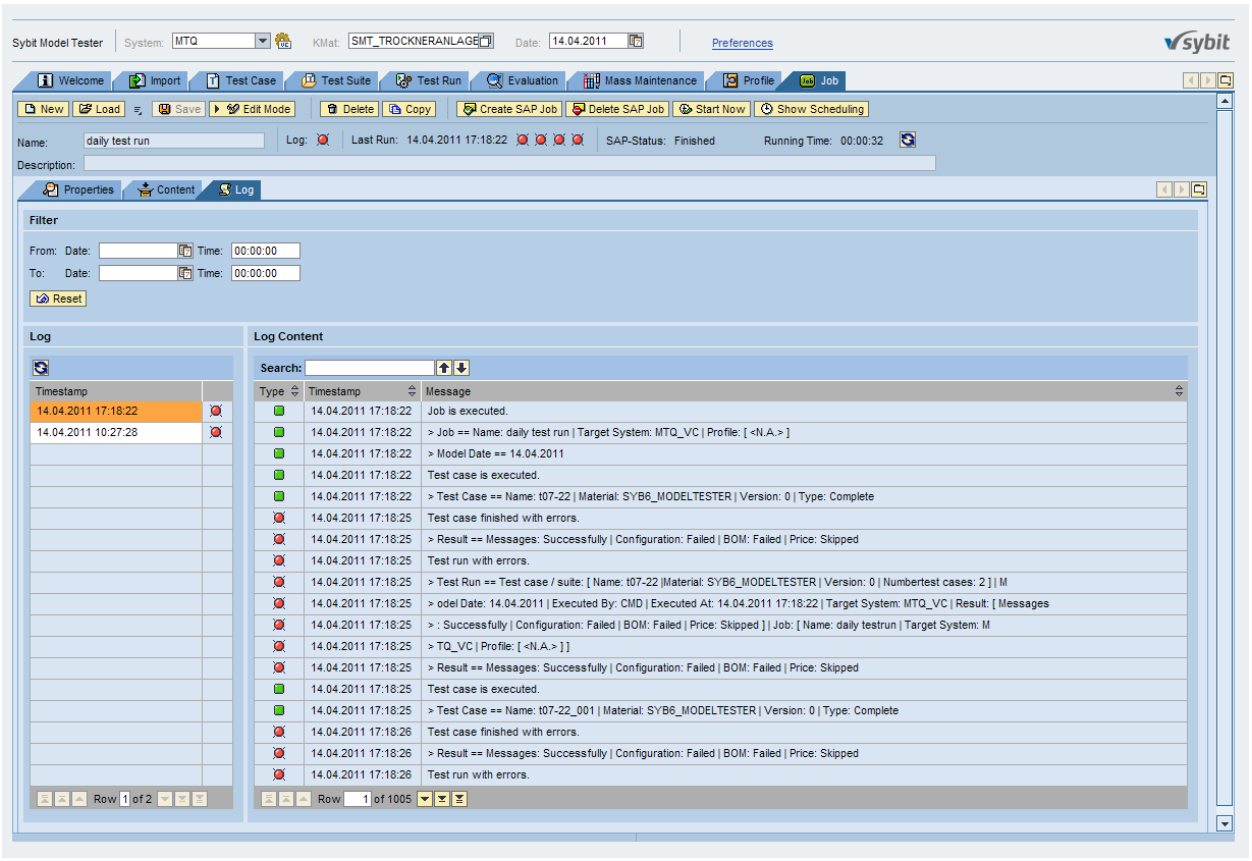

Sybit Model Tester
System: MTQ
KMat: SMT_TROCKNERANLAGE
Date: 14.04.2011
Preferences
Welcome
Import
Test Case
Test Suite
Test Run
Evaluation
Mass Maintenance
Profile
Job
New
Load
Save
Edit Mode
Delete
Copy
Create SAP Job
Delete SAP Job
Start Now
Show Scheduling
Name: daily test run
Log:
Last Run: 14.04.2011 17:18:22
SAP-Status: Finished
Running Time: 00:00:32
Description:
Properties
Content
Log
Filter
From: Date: Time: 00:00:00
To: Date: Time: 00:00:00
Reset
Log
Log Content
Search:
Timestamp
14.04.2011 17:18:22
14.04.2011 10:27:28
Row 1 of 2
Type
Timestamp
Message
14.04.2011 17:18:22 Job is executed.
14.04.2011 17:18:22 > Job == Name: daily test run | Target System: MTQ_VC | Profile: [<N.A.>]
14.04.2011 17:18:22 > Model Date == 14.04.2011
14.04.2011 17:18:22 Test case is executed.
14.04.2011 17:18:22 > Test Case == Name: t07-22 | Material: SYB6_MODELTESTER | Version: 0 | Type: Complete
14.04.2011 17:18:25 Test case finished with errors.
14.04.2011 17:18:25 > Result == Messages: Successfully | Configuration: Failed | BOM: Failed | Price: Skipped
14.04.2011 17:18:25 Test run with errors.
14.04.2011 17:18:25 > Test Run == Test case / suite: [Name: t07-22 |Material: SYB6_MODELTESTER | Version: 0 | Numbertest cases: 2] | M
14.04.2011 17:18:25 > odel Date: 14.04.2011 | Executed By: CMD | Executed At: 14.04.2011 17:18:22 | Target System: MTQ_VC | Result: [Messages
14.04.2011 17:18:25 > : Successfully | Configuration: Failed | BOM: Failed | Price: Skipped] | Job: [Name: daily testrun | Target System: M
14.04.2011 17:18:25 > TQ_VC | Profile: [<N.A.>]]
14.04.2011 17:18:25 > Result == Messages: Successfully | Configuration: Failed | BOM: Failed | Price: Skipped
14.04.2011 17:18:25 Test case is executed.
14.04.2011 17:18:25 > Test Case == Name: t07-22_001 | Material: SYB6_MODELTESTER | Version: 0 | Type: Complete
14.04.2011 17:18:26 Test case finished with errors.
14.04.2011 17:18:26 > Result == Messages: Successfully | Configuration: Failed | BOM: Failed | Price: Skipped
14.04.2011 17:18:26 Test run with errors.
Row 1 of 1005

Fig. 4-52: Job tab: Log tab.

4.15 Saving Test Cases from CU50

As an optional extension to the Sybit Model Tester, there is a plugin for transaction CU50 that allows you to save test cases in the Sybit Model Tester. To find out if this plugin is installed in your system, please contact your system administration. In the Sybit Model Tester standard software, the transaction CU50 is not modified. In contrast, we ship a transaction Y_CU50 that contains the additional saving functionality. Apart from the saving functionality, there is no difference between CU50 and Y_CU50 from a visual viewpoint.

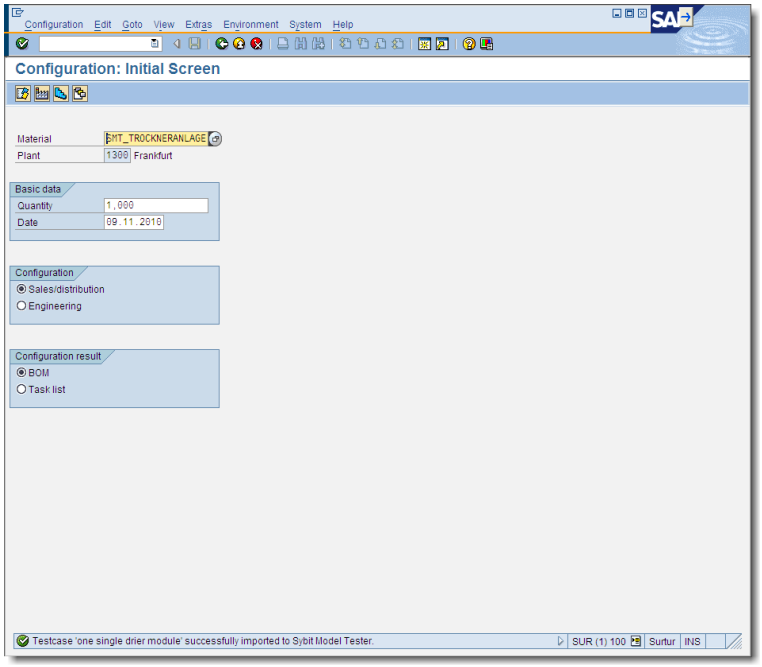


Fig. 4-53: Start screen of Y_CU50.

To be able to save the values from Y_CU50 correctly in the Sybit Model Tester you need to set the following parameters in the Y_CU50 (in the menu at **View → Settings**).

- **Default Values**: Copy
- **Configurator**: Permanent

If those parameters are set differently then in some cases characteristic values that are set by dependencies are not saved into the expected result but in the input configuration.

To trigger the save function you just have to click on the save button in the menu bar. In the standard transaction CU50 this button is disabled. If you enter a complete configuration and press save, CU50 is left and a popup dialogue appears that prompts you to enter a test case name.

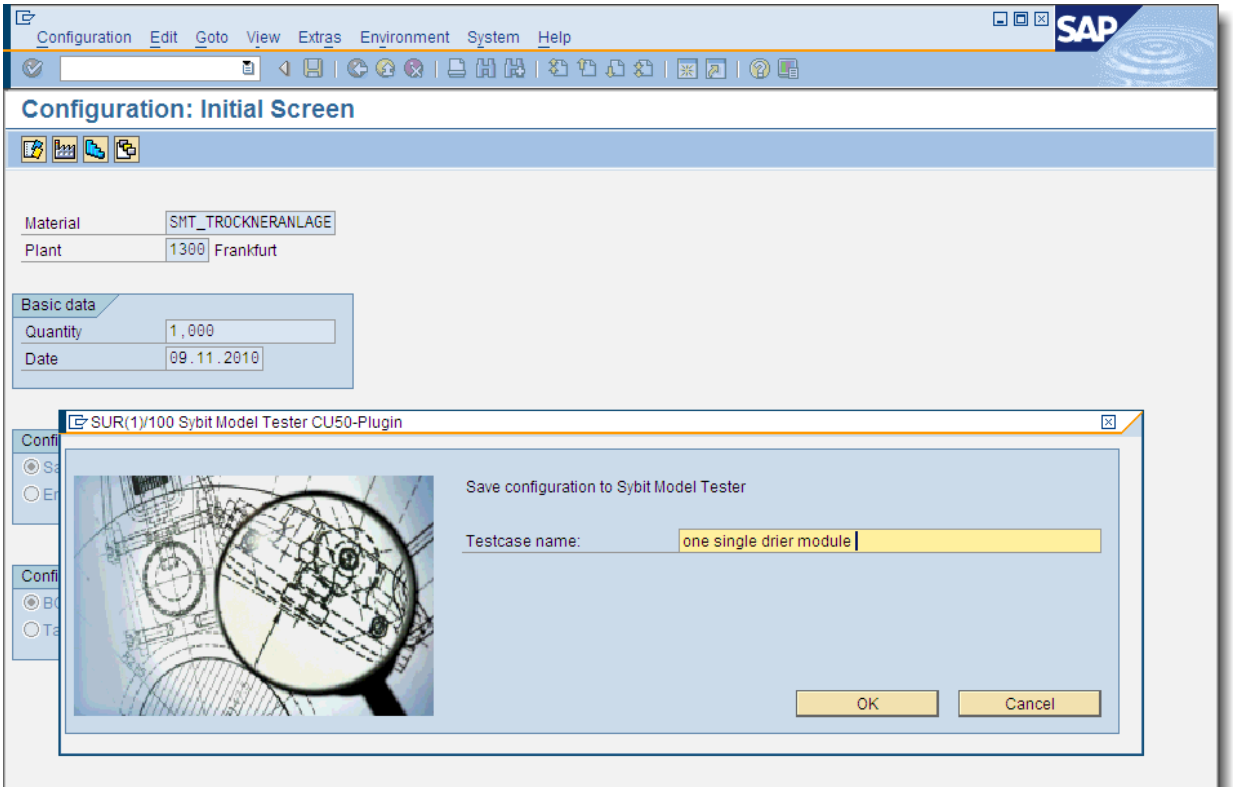


Fig. 4-54: Popup dialogue for saving a configuration as test case in the Sybit Model Tester: Input of test case name.

If you enter a test case name and press OK, the test case is saved in the Sybit Model Tester. If you press Cancel, your previous configuration will be discarded.

5 Error Handling

This section tells you how you should deal with application errors.

But what is an application error? In principle there are two kinds of errors: Errors that terminate the application (so called dumps) and logical errors that cause the application to behave differently as described in this manual.

5.1 Termination (Dump) of the Application

Fig. 5-1 shows an example of an error that terminates the application. If this occurs, the Sybit Model Tester makes sure that the underlying database is always consistent. To this end, all changes that have been entered after the last click on a save button are dismissed.

Please proceed as follows:

1. Retrieve the system context by clicking on the link **Retrieve context** and save it to your local computer, e.g. your desktop. The saved information is important for error analysis.
2. Open a new email message and address it to support.smt@sybit.de .
3. Append the saved file as an attachment to your email.
4. Note down the steps that caused the error, e.g.: *Loaded test case <NAME> → assigned value <VALUE> to characteristic <CHARACTERISTIC> → saved the test → jumped to Test Run by Quick Link → error occurred.* Please attach this text also to the Email.
5. Restart the Sybit Model Tester by clicking F5.

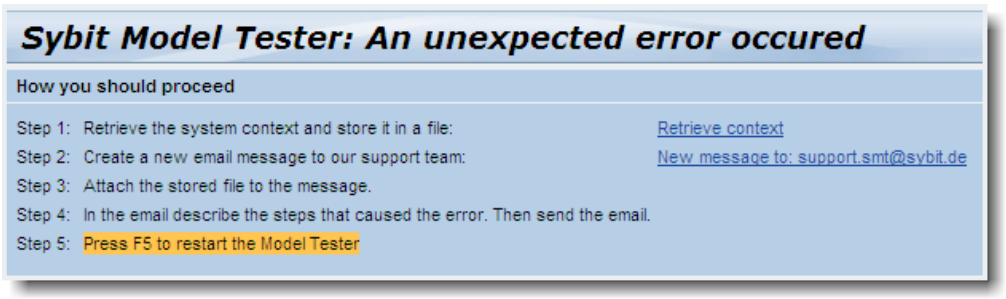

Sybit Model Tester: An unexpected error occured
How you should proceed
Step 1: Retrieve the system context and store it in a file:
Retrieve context
Step 2: Create a new email message to our support team:
New message to: support.smt@sybit.de
Step 3: Attach the stored file to the message.
Step 4: In the email describe the steps that caused the error. Then send the email.
Step 5: Press F5 to restart the Model Tester

Fig. 5-1: Termination of the application.

5.2 Wrong Functionality

An example for wrong functionality could be that before saving a test case a description has been defined. However, on load of the test, this description is not displayed any more.

Proceed as follows:

1. Try to reproduce the behavior, i.e. find out if you can produce the error again and again. To track the error it is helpful, if your reproduction contains as few steps as possible.
2. Note down the steps of your reproduction, e.g.: *Loaded test case <NAME> → assigned value <VALUE> to description → saved the test → edit off → loaded test → description is not set*. Insert this reproduction instruction into an email.

3. Send the Email to support.smt@sybit.de.

Appendix

A. Messages

The tables in this appendix contain all possible messages that may be contained in the messages tab of the expected or factual result of a test case.

A.1. VC-Specific Messages

The following messages are returned by the VC and adopted by the Sybit Model Tester.

Message
Invalid currency
Invalid date
Invalid time
Invalid format
Invalid interval
Interval forbidden
Invalid pattern
Invalid unit of measure
Invalid base measure
Invalid size
Value not found
Value not possible
Change rejected
Value inconsistent
Characteristic univalent
Characteristic not selectable
Characteristic not found
Characteristic locked
Object characteristic
Characteristic not assigned
Characteristic inconsistent
Restricted value domain
Configuration inconsistent
Unknown error
Input too long
Unknown error

A.2. Errors in the Test Case Structure

The following error messages are triggered, if the structure made from header KMat and sub KMats of a multi level test case is inconsistent with the structure proposed by the VC.

Message	Description
Structure conflict	(Deprecated: Not used any more)
Obsolete test structure	The structure of the test case contains a sub KMat that is not contained in the structure retrieved from the VC.
Test structure not found	The structure of the test cases lacks a sub KMat that has been part of the structure returned by the VC.

A.3. Error Messages Caused by Testing Prices

The following error messages are triggered by testing prices in a test run.

Message	Description
Condition not found	For the variant condition and the calculation scheme in the header of the test case, there exists no condition record in all access sequences of all access.
Access sequence inconsistent	For the given variant condition, the system found a condition record in an access of a sequence. However, the variant condition is not found in the last access of the same sequence.
Access to condition not found	(deprecated)

For more information about theses messages, please refer to the section on testing prices (Sec. 2.3) and the section on expected prices in the test case tab (Sec. 4.8.8).

B. Further Information

SMT 2.2

Customer area on the web:	Customer Area
Public web page Sybit Model Tester:	www.sybitmodeltester.de
Web page Sybit GmbH:	www.sybit.com
Email address for support inquiries:	support.smt@sybit.de